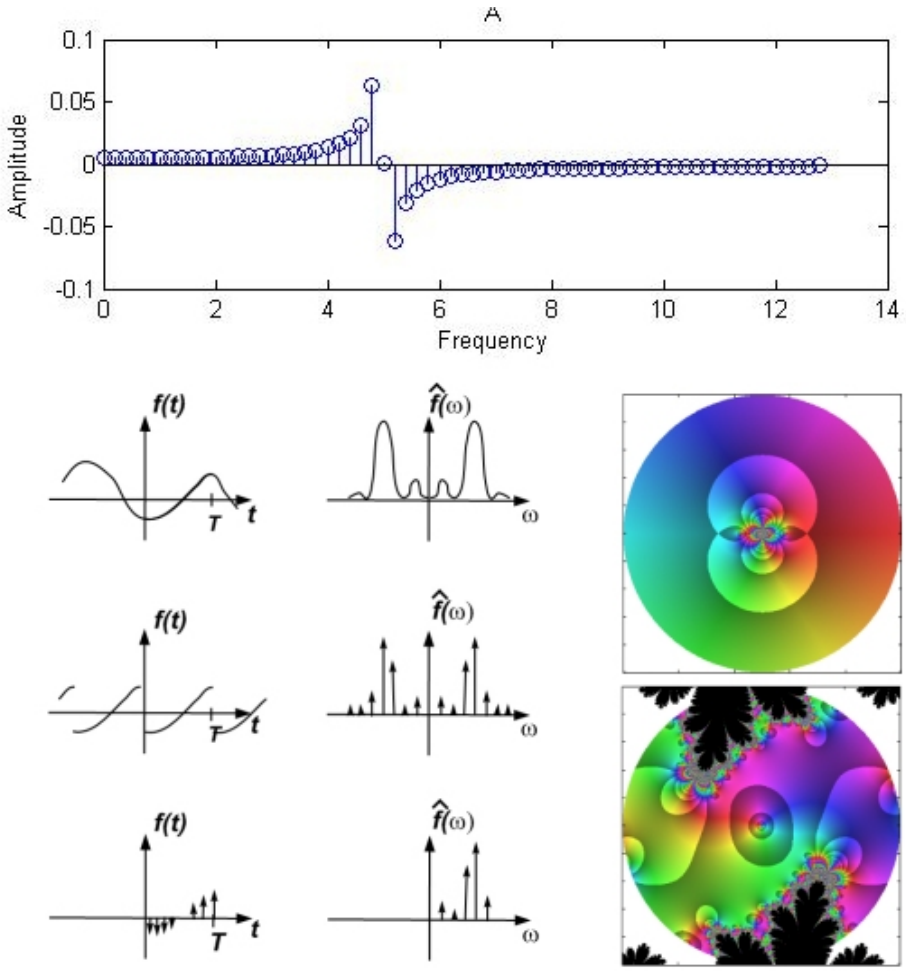
RUSSELL L. HERMAN

AN INTRODUCTION TO FOURIER ANALYSIS



R. L. HERMAN - VERSION DATE: MARCH 6, 2016

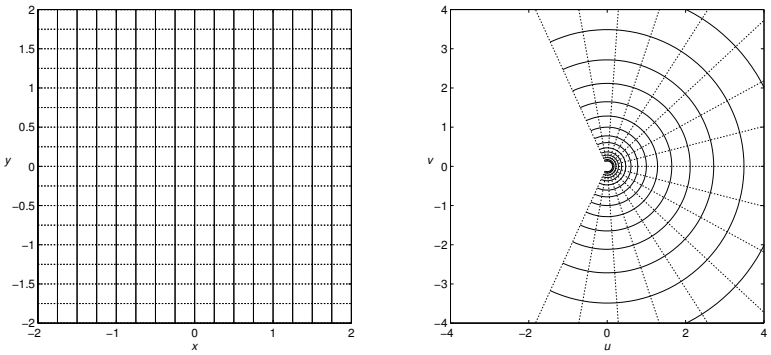


Figure 4.7: 2D plot showing how the function $f(z) = e^z$ maps a grid in the z-plane into the w-plane.

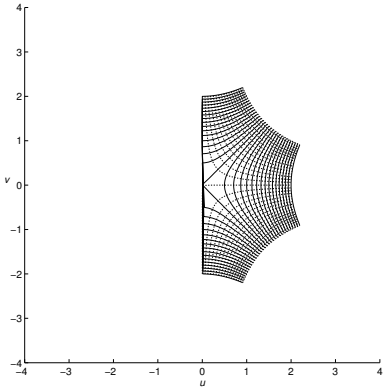


Figure 4.8: 2D plot showing how the function $f(z) = \sqrt{z}$ maps a grid in the z-plane into the w-plane.

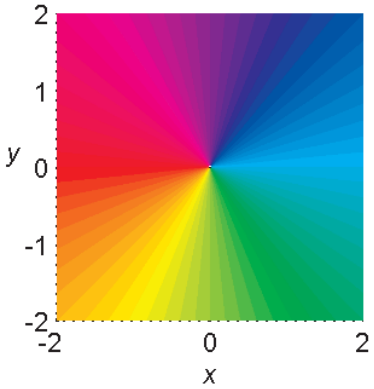


Figure 4.9: Domain coloring of the complex z-plane assigning colors to $\arg(z)$.

Example 4.7. Find the real and imaginary parts of $f(z) = \ln z$.

In this case, we make use of the polar form of a complex number, $z = re^{i\theta}$. Our first thought would be to simply compute

$$\ln z = \ln r + i\theta.$$

However, the natural logarithm is multivalued, just like the square root function. Recalling that $e^{2\pi ik} = 1$ for k an integer, we have $z = re^{i(\theta+2\pi k)}$. Therefore,

$$\ln z = \ln r + i(\theta + 2\pi k), \qquad k = \text{ integer.}$$

The natural logarithm is a multivalued function. In fact, there are an infinite number of values for a given z. Of course, this contradicts the definition of a function that you were first taught.

Thus, one typically will only report the principal value,

$$\operatorname{Log} z = \ln r + i\theta,$$

for θ restricted to some interval of length 2π, such as $[0, 2\pi)$. In order to account for the multivaluedness, one introduces a way to extend the complex plane so as to include all of the branches. This is done by assigning a plane to each branch, using (branch) cuts along lines, and then gluing the planes together at the branch cuts to form what is called a Riemann surface. We will not elaborate upon this any further here and refer the interested reader to more advanced texts. Comparing the multivalued logarithm to the principal value logarithm, we have

$$\ln z = \operatorname{Log} z + 2n\pi i.$$

We should note that some books use $\log z$ instead of $\ln z$. It should not be confused with the common logarithm.

4.2.1 *Complex Domain Coloring*

ANOTHER METHOD FOR VISUALIZING COMPLEX FUNCTIONS is domain coloring. The idea was described by Frank A. Farris. There are a few ap-

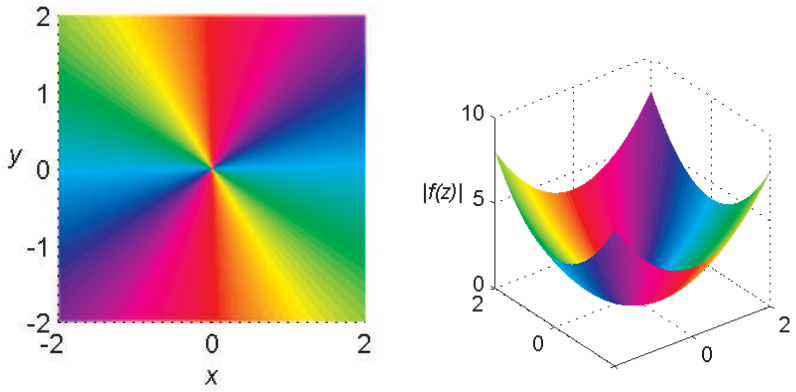


Figure 4.10: Domain coloring for $f(z) = z^2$. The left figure shows the phase coloring. The right figure shows the colored surface with height $|f(z)|$.

proaches to this method. The main idea is that one colors each point of the z-plane (the domain) according to $\arg(z)$ as shown in Figure 4.9. The modulus, $|f(z)|$ is then plotted as a surface. Examples are shown for $f(z) = z^2$ in Figure 4.10 and $f(z) = 1/z(1-z)$ in Figure 4.11.

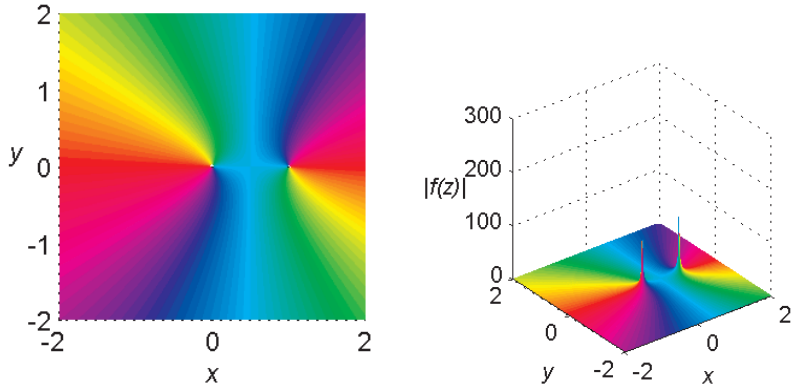


Figure 4.11: Domain coloring for $f(z) = 1/z(1-z)$. The left figure shows the phase coloring. The right figure shows the colored surface with height $|f(z)|$.

We would like to put all of this information in one plot. We can do this by adjusting the brightness of the colored domain using the modulus of the function. In the plots that follow we use the fractional part of $\ln|z|$. In Figure 4.12 we show the effect for the z-plane using $f(z) = z$. In the figures that follow, we look at several other functions. In these plots, we have chosen to view the functions in a circular window.

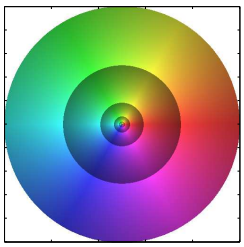


Figure 4.12: Domain coloring for the function $f(z) = z$ showing a coloring for $\arg(z)$ and brightness based on $|f(z)|$.

One can see the rich behavior hidden in these figures. As you progress in your reading, especially after the next chapter, you should return to these figures and locate the zeros, poles, branch points, and branch cuts. A search online will lead you to other colorings and superposition of the uv grid on these figures.

As a final picture, we look at iteration in the complex plane. Consider the function $f(z) = z^2 - 0.75 - 0.2i$. Interesting figures result when studying the iteration in the complex plane. In Figure 4.15 we show $f(z)$ and $f^{20}(z)$, which is the iteration of f twenty times. It leads to an interesting coloring. What happens when one keeps iterating? Such iterations lead to the study of Julia and Mandelbrot sets . In Figure 4.16 we show six iterations of

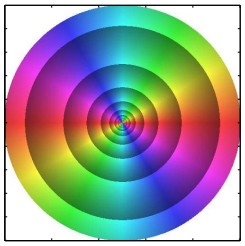


Figure 4.13: Domain coloring for the function $f(z) = z^2$.

$$f(z) = (1 - i/2)\sin x.$$

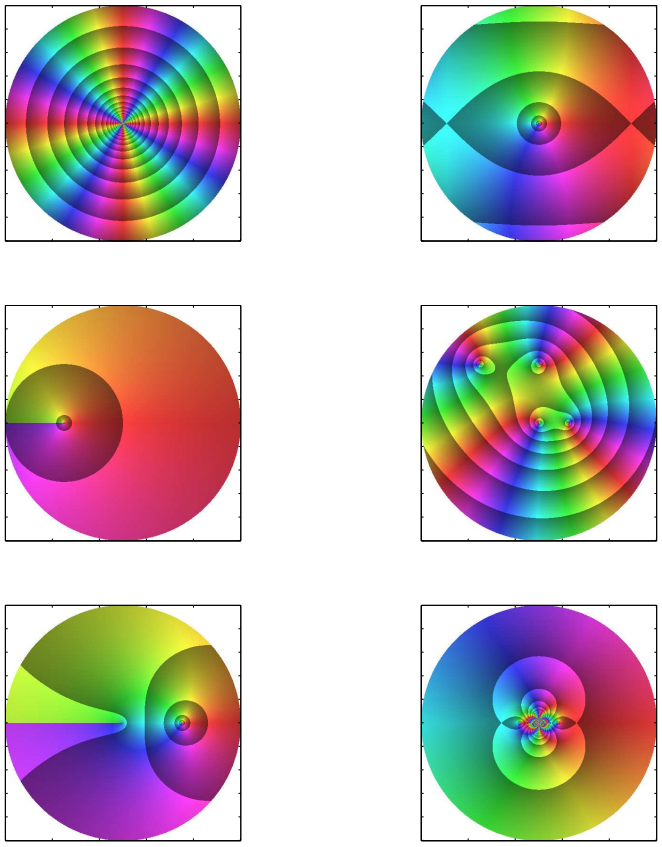


Figure 4.14: Domain coloring for several functions. On the top row, the domain coloring is shown for $f(z) = z^4$ and $f(z) = \sin z$. On the second row, plots for $f(z) = \sqrt{1+z}$ and $f(z) = \frac{1}{z(1/2-z)(z-i)(z-i+1)}$ are shown. In the last row, domain colorings for $f(z) = \ln z$ and $f(z) = \sin(1/z)$ are shown.

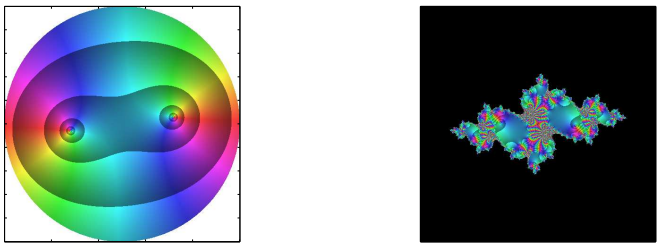


Figure 4.15: Domain coloring for $f(z) = z^2 - 0.75 - 0.2i$. The left figure shows the phase coloring. On the right is the plot for $f^{20}(z)$.

The following code was used in MATLAB to produce these figures.

```
fn = @(x) (1-i/2)*sin(x);
xmin=-2; xmax=2; ymin=-2; ymax=2;
Nx=500;
Ny=500;
x=linspace(xmin,xmax,Nx);
y=linspace(ymin,ymax,Ny);
[X,Y] = meshgrid(x,y); z = complex(X,Y);
tmp=z; for n=1:6
    tmp = fn(tmp);
```

```
end Z=tmp;
XX=real(Z);
YY=imag(Z);
R2=max(max(X.^2));
R=max(max(XX.^2+YY.^2));

circle(:,:,1) = X.^2+Y.^2 < R2;
circle(:,:,2)=circle(:,:,1);
circle(:,:,3)=circle(:,:,1);

addcirc(:,:,1)=circle(:,:,1)==0;
addcirc(:,:,2)=circle(:,:,1)==0;
addcirc(:,:,3)=circle(:,:,1)==0;

warning off MATLAB:divideByZero;
hsvCircle=ones(Nx,Ny,3);
hsvCircle(:,:,1)=atan2(YY,XX)*180/pi+(atan2(YY,XX)*180/pi<0)*360;
hsvCircle(:,:,1)=hsvCircle(:,:,1)/360; lgz=log(XX.^2+YY.^2)/2;
hsvCircle(:,:,2)=0.75; hsvCircle(:,:,3)=1-(lgz-floor(lgz))/2;
hsvCircle(:,:,1) = flipud((hsvCircle(:,:,1)));

hsvCircle(:,:,2) = flipud((hsvCircle(:,:,2)));

hsvCircle(:,:,3) =flipud((hsvCircle(:,:,3)));

rgbCircle=hsv2rgb(hsvCircle);
rgbCircle=rgbCircle.*circle+addcirc;

image(rgbCircle)
axis square
set(gca,'XTickLabel',{})
set(gca,'YTickLabel',{})
```

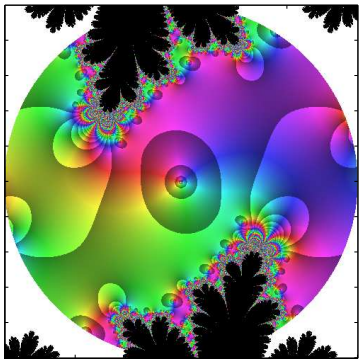
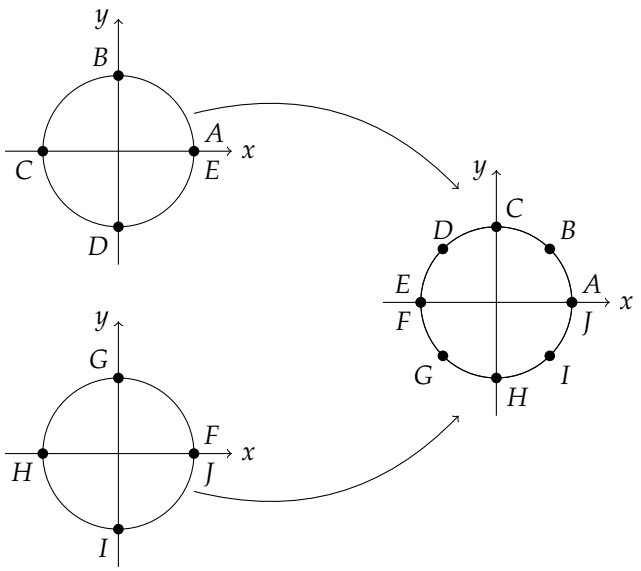


Figure 4.16: Domain coloring for six iterations of $f(z) = (1 - i/2)\sin x$.

Figure 4.50: In this figure we show the combined mapping using two branches of the square root function.



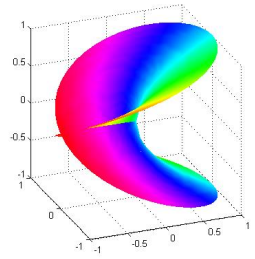


Figure 4.51: Riemann surface for $f(z) = z^{1/2}$.

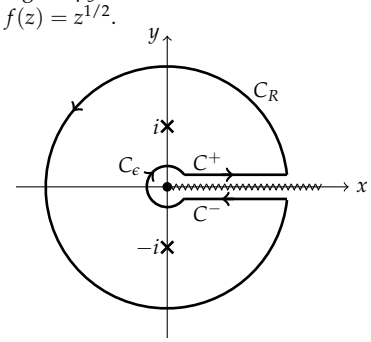


Figure 4.52: An example of a contour which accounts for a branch cut.

We now look at examples involving integrals of multivalued functions.

Example 4.45. Evaluate $\displaystyle\int_0^\infty \frac{\sqrt{x}}{1+x^2}\,dx$.

We consider the contour integral $\oint_C \frac{\sqrt{z}}{1+z^2}dz$. The first thing we can see in this problem is the square root function in the integrand. Being that there is a multivalued function, we locate the branch point and determine where to draw the branch cut. In Figure 4.52 we show the contour that we will use in this problem. Note that we picked the branch cut along the positive x-axis.

We take the contour C to be positively oriented, being careful to enclose the two poles and to hug the branch cut. It consists of two circles. The outer circle C_R is a circle of radius R and the inner circle C_ϵ will have a radius of ϵ. The sought-after answer will be obtained by letting $R \to \infty$ and $\epsilon \to 0$. On the large circle we have that the integrand goes to zero fast enough as $R \to \infty$. The integral around the small circle vanishes as $\epsilon \to 0$. We can see this by parametrizing the circle as $z = \epsilon e^{i\theta}$ for $\theta \in [0, 2\pi]$:

$$\begin{aligned}\oint_{C_\epsilon} \frac{\sqrt{z}}{1+z^2}dz &= \int_0^{2\pi} \frac{\sqrt{\epsilon e^{i\theta}}}{1+(\epsilon e^{i\theta})^2} i\epsilon e^{i\theta} d\theta \\ &= i\epsilon^{3/2}\int_0^{2\pi} \frac{e^{3i\theta/2}}{1+(\epsilon^2 e^{2i\theta})} d\theta. \end{aligned} \tag{4.69}$$

It should now be easy to see that as $\epsilon \to 0$, this integral vanishes.

The integral above the branch cut is the one we are seeking,

$$\lim_{\substack{R\to\infty \\ \epsilon\to 0}} \int_{C_+} \frac{\sqrt{z}}{1+z^2}\,dz = \int_0^\infty \frac{\sqrt{x}}{1+x^2}\,dx.$$

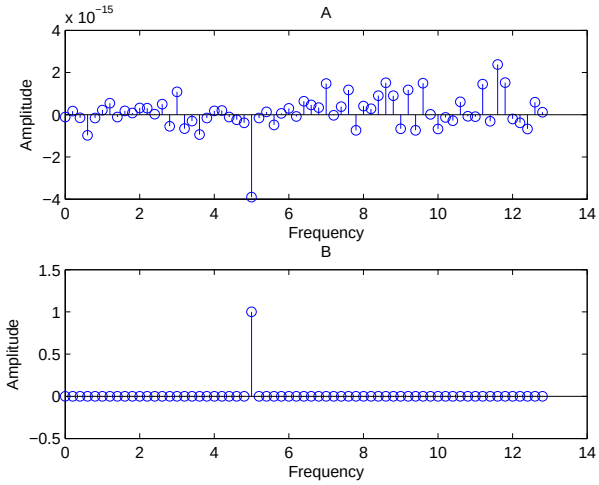


Figure 6.13: Computed discrete Fourier coefficients for $y(t) = \sin(10\pi t)$, with $N = 128$ points on the interval $[0, 5]$.

discrete Fourier series, we obtain a curve representing the sampled signal. This is shown in Figure 6.14. Note that the sampled data is represented as data points (circles) and the reconstructed signal is function displayed on the plot.

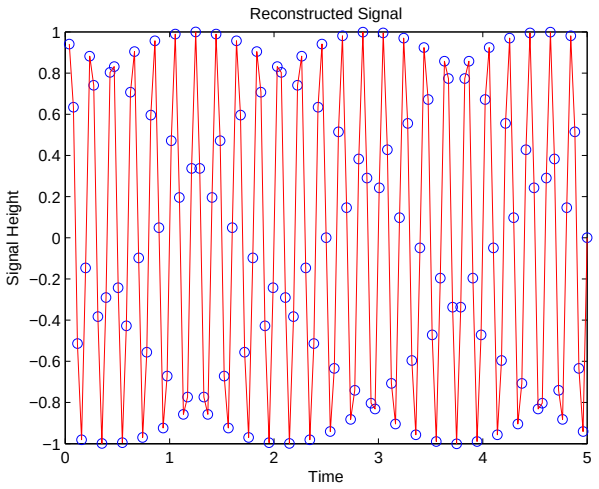


Figure 6.14: Reconstruction of $y(t) = \sin(10\pi t)$ from its Fourier coefficients.

We can look at more interesting functions. For example, we can add two pure notes together or make the amplitude time depdentent, such as shown in the next examples.

Example 6.12. Determine the frequency content of the signal

$$y(t) = \sin(10\pi t) - \frac{1}{2}\cos(6\pi t).$$

This is the sum of two notes with frequencies of 5 Hz and 3 Hz. It also shows a mixing of sines and cosines. The occurance of a sine function does not affect the frequency content, as the sine function is just a shift cosine function.

Sampling this signal with $N = 128$ points on the interval $[0, 5]$, we see from Figure 6.15 that the implementation picks out the correct frequencies and amplitudes.

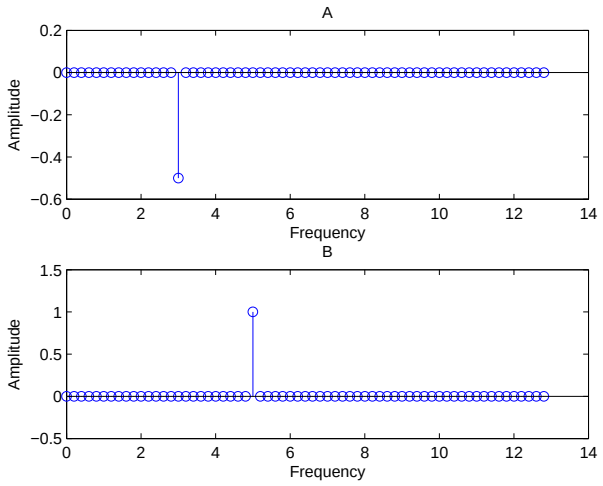


Figure 6.15: Computed discrete Fourier coefficients for $\sin(10\pi t) - \frac{1}{2}\cos(6\pi t)$ with $N = 128$ points on the interval $[0, 5]$.

Example 6.13. Find the Fourier coefficients for the signal

$$y(t) = e^{-\alpha t}\sin(10\pi t).$$

In this example we investigate the Fourier coefficients for a slightly more complicated signal, one with a decaying amplitude. This could represent a damped oscillator.

For $\alpha = 0.1$, we consider the signal

$$y(t) = e^{-0.1t}\sin(10\pi t).$$

Here we again sample the signal with $N = 128$ points on the interval $[0, 5]$. The Fourier coefficients are shown in Figure 6.16 and the corresponding reconstruction is shown in Figure 6.17. In this case we see that a number of modes are excited and we no longer have one frequency.

We will look into more interesting features in discrete signals in the next chapter. We end with an application to a small data set and leave the analysis of this and other signals as an exercise for the reader.

Example 6.14. Analysis of monthly mean surface temperatures.

Consider the data[2] of monthly mean surface temperatures at Amphitrite Point, Canada shown in Table 6.2. The temperature was recorded in ^{0}C and averaged for each month over a two year period. We would like to look for the frequency content of this time series.

[2] This example is from *Data Analysis Methods in Physical Oceanography*, W. J. Emery and R.E. Thomson, Elsevier, 1997.

Month	1	2	3	4	5	6	7	8	9	10	11	12
1982	7.6	7.4	8.2	9.2	10.2	11.5	12.4	13.4	13.7	11.8	10.1	9.0
1983	8.9	9.5	10.6	11.4	12.9	12.7	13.9	14.2	13.5	11.4	10.9	8.1

Table 6.2: Monthly mean surface temperatures (^{0}C) at Amphitrite Point, Canada for 1982-1983.

In Figure 6.18 we plot the data in Table 6.2 as circles. We then use the data to compute the Fourier coefficients. These coefficients are

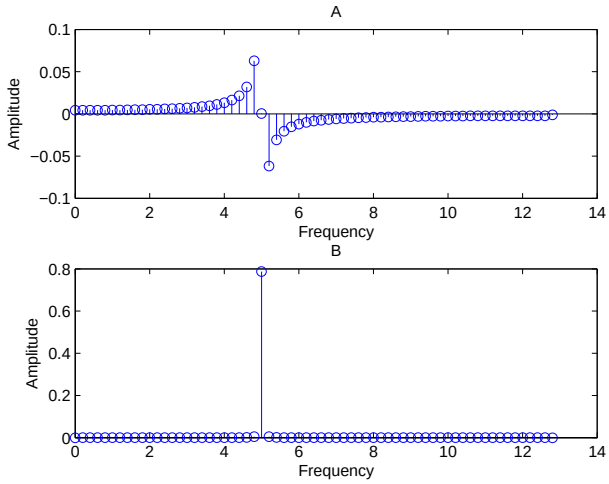


Figure 6.16: Computed discrete Fourier coefficients for $y(t) = e^{-\alpha t}\sin(10\pi t)$ with $\alpha = 0.1$ and $N = 128$ points on the interval $[0, 5]$.

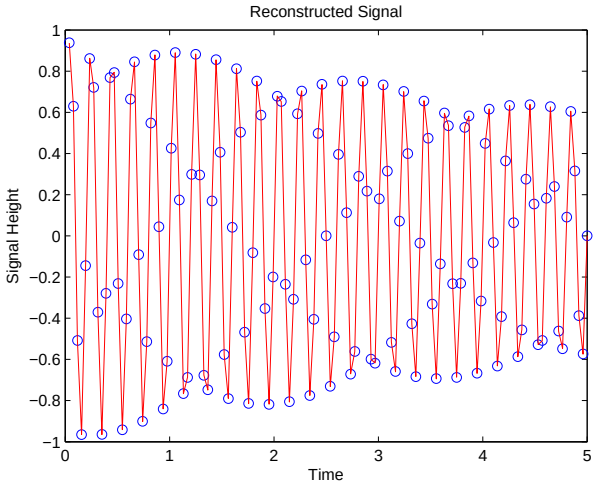


Figure 6.17: Reconstruction of $y(t) = e^{\alpha t}\sin(10\pi t)$ with $\alpha = 0.1$ from its Fourier coefficients.

inserted into the discrete Fourier series and plotted as a curve. We see that the resulting reconstruction fits the data.

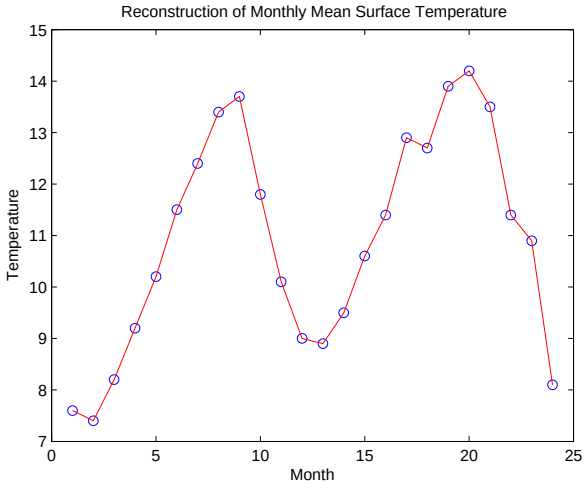


Figure 6.18: Plot and reconstruction of the monthly mean surface temperature data.

6.8 MATLAB Implementation

Discrete Fourier Transforms and FFT are easily implemented in computer applications. In this section we describe the MALAB routines used in this book.

6.8.1 MATLAB for the Discrete Fourier Transform

In this section we provide implementations of the discrete trigonometric transform in MATLAB. The first implementation is a straightforward one which can be done in most programming languages. The second implementation makes use of matrix computations that can be performed in MATLAB or similar programs like GNU Octave or Pylab. Sums can be done with matrix multiplication, as described in the next section. This eliminates the loops in the first program below and speeds up the computation for large data sets.

Direct Implementation for a data set

The following code was used to produce Figure 6.18. It shows a direct implementation using loops to compute the trigonometric DFT as developed in this chapter. The data is entered in vector $\mathbf{y}$. The Fourier coefficients are entered using the matrix capabilities of MATLAB as described in the next section. The signal is then reconstructed using the finite series representation. Plots are provided to show this implementations as demonstrated in earlier examples.

Chapter 7

Signal Analysis

There's no sense in being precise when you don't even know what you're talking about. - John von Neumann (1903 - 1957)

7.1 Introduction

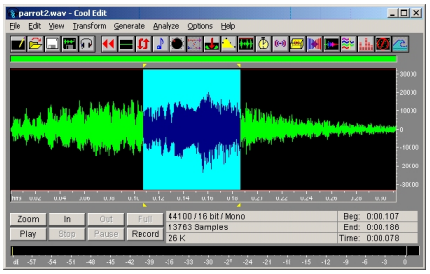


Figure 7.1: Cool Edit displaying a WAV file and its properties.

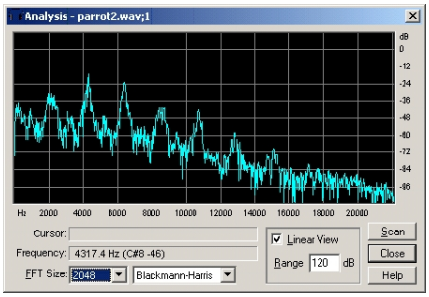


Figure 7.2: Cool Edit displaying the spectrum of a WAV file.

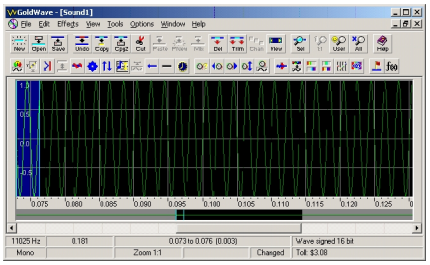


Figure 7.3: Goldwave displaying a simple tone.

It is now time to look back at the Introducton and see what it was that we promised to do in this course. The goal was to develop enough tools to begin to understand what happens to the spectral content of signals when analog signals are discretized. We started with a study of Fourier series and just ended with discrete Fourier transforms. We have seen how Fourier transform pairs, $f(t)$ and $\hat{f}(\omega)$, are affected by recording a signal over a finite time T at a sampling rate of f_s. This in turn lead to the need for discrete Fourier transforms (DFTs). However, we have yet to see some of the effects of this discretization on the information that we obtain from the spectral analysis of signals in practice. In this chapter we will look at results of applying DFTs to a variety of signals.

The simplest application of this analysis is the analysis of sound. Music, which is inherently an analog signal, is recorded over a finite time interval and is sampled at a rate that yields pleasing sounds that can be listened to on the computer, a CD, or in an MP3 player.

You can record and edit sounds yourself. There are many audio editing packages that are available. We have successfully used these packages plus some minimal applets and mathematics packages to introduce high school students and others with a minimal mathematics background to the Fourier analysis of sounds. As we have seen, we need only understand that signals can be represented as sums of sinusoidal functions of different frequencies and amplitudes.

For example, we have had students working with musical instruments, bird sounds, dolphin sounds, ECGs, EEGs, digital images, and other forms of recorded signals or information. One just needs to find a way to determine the frequency content of the signal and then pick out the dominant frequencies to reconstruct the signal.

There are many packages that can be used to display sound and time series. We will see how to use MATLAB, although one could use Maple or

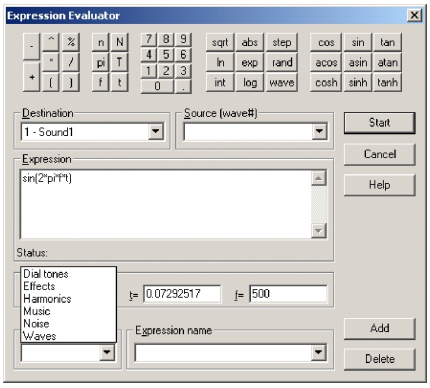


Figure 7.4: Goldwave display of function editor.

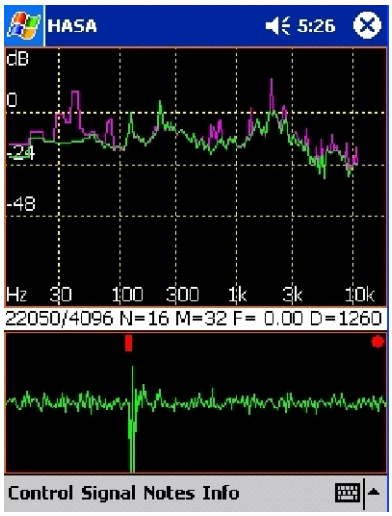


Figure 7.5: HASA for pocket PCs.

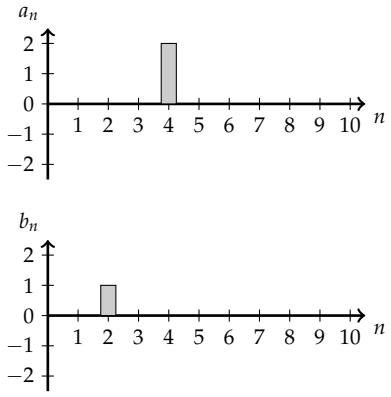


Figure 7.6: This Figure shows the spectral coefficients for a signal of the form $f(t) = 2\cos 4t + \sin 2t$.

Mathematica to import and analyze sounds. There are also stand alone editors like Cool Edit Pro (bought out by Adobe in 2003 and renamed Adobe Audition), Audacity (http://audacity.sourceforge.net/) an open source editor, or Goldwave (http://www.goldwave.com/), which allows one to input a formula and "play" it.

A sample Cool Edit session is shown in Figure 7.1. In this figure is displayed the sound print of a parrot. It is obviously a complex signal, made up of many harmonics. Highlighting a part of the signal, one can look at the frequency content of this signal. Such a spectrum is shown in Figure 7.2. Notice the spikes every couple of thousand Hertz.

Not many fancy, but inexpensive, sound editors have a frequency analysis component like Cool Edit had. One has to go out on to the web and search for features that do not just entail editing sounds for MP3 players. Goldwave allows one to enter a formula and then listen to the corresponding sounds. This is also a feature not found in most editors. However, it is a useful tool that takes little "programming" to connect the mathematics to the signal. Cool Edit and others have a feature to generate tones, but this is more exact. The interface for Goldwave is shown in Figure 7.3 and the function editor is in 7.4. However, there are plenty of other editors. In the early 2000's the HASA (Handheld Audio Spectrum Analyzer) application shown in Figure 7.5 was a good tool for pocket PCs. Also, spectrum analyzers are available for mobile devices, such as the iPhone (e.g. Pocket RTA - Spectrum Analyser).

7.2 *Periodogram Examples*

THE NEXT STEP IN THE ANALYSIS IS TO UNDERSTAND the output of the discrete transform, or the Fast Fourier Transform (FFT), that is generated by such programs. Often we see spectrograms or periodograms. We should understand what it is that they produce. As an example, lets say we have the sum of a sine and a cosine function with different frequencies and amplitudes. We could represent the discrete Fourier coefficients as a_n's and b_n's, like we have computed many times in the course, in simple plots of the coefficients vs n (or the frequency) such as shown in Figure 7.6. In this case there is a cosine contribution of amplitude two at frequency $f = 4$ Hz and a sine contribution of amplitude one at frequency $f = 2$ Hz. It takes two plots to show both the a_n's and b_n's. However, we are often only interested in the energy content at each frequency. For this example, the last plot in Figure 7.7 shows the spectral content in terms of the modulus of the signal.

For example, $\cos 5t$ and $3\sin 5t$ would have spikes in their respective plots at the same frequency, $f = \frac{5}{2\pi}$. As noted earlier in Equation (6.30), we can write the sum $\cos 5t + 3\sin 5t$ as a single sine function with an amplitude $c_n = \sqrt{a_n^2 + b_n^2}$. Thus, a plot of the "modulus" of the signal is used more often. However, in the examples we will sometimes display both forms to bring home the relationship between the trigonometric and exponential

forms of the Fourier spectrum of the signal.

Once one has determined the Fourier coefficients, then one can reconstruct the signal. In the case that one has the exact components, then the reconstruction should be perfect as shown for the previous example in Figure 7.6. The reconstruction in this case gave the plot in Figure 7.8. However, for real signals one does not know ahead of time what the actual frequencies are that made up the signal.

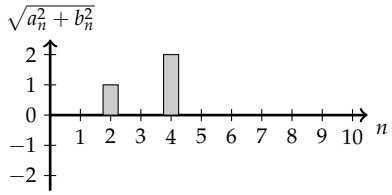


Figure 7.7: This Figure shows the spectrum for a signal of the form $f(t) = 2\cos 4t + \sin 2t$.

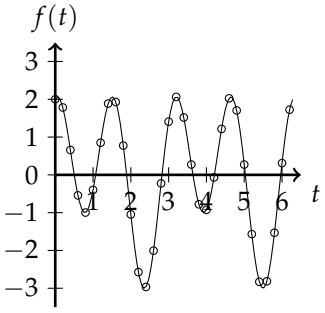


Figure 7.8: This Figure shows the original signal of the form $f(t) = 2\cos 4t + \sin 2t$ and a reconstruction based upon the series expansion.

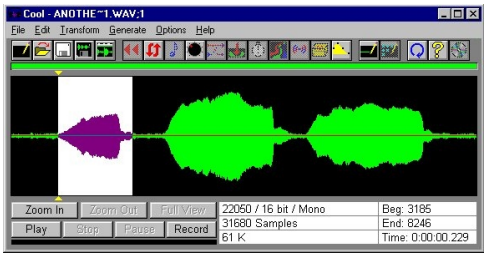


Figure 7.9: A piece of a typical bird sound.

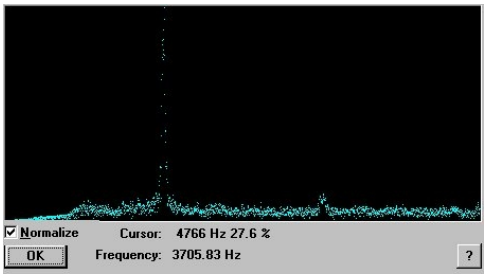


Figure 7.10: A Fourier analysis of the bird sound in Figure 7.9.

For example, one could analyze a bird sound like the one shown in Figure 7.9. We capture a part of the sound and look at its spectrum. An example is shown in Figure 7.10. Notice that the spectrum is not very clean, although a few peaks stand out. We had a group of high school students carry out this procedure. The students picked out a few of the dominant frequencies and the corresponding amplitudes. Using just a few frequencies, they reconstructed the bird signals. In Figure 7.11 we show the original and reconstructed signals, respectively. While these might not look exactly the same, they do sound very similar.

There are different methods for displaying the Fourier spectrum of signals. Here we define some of these.

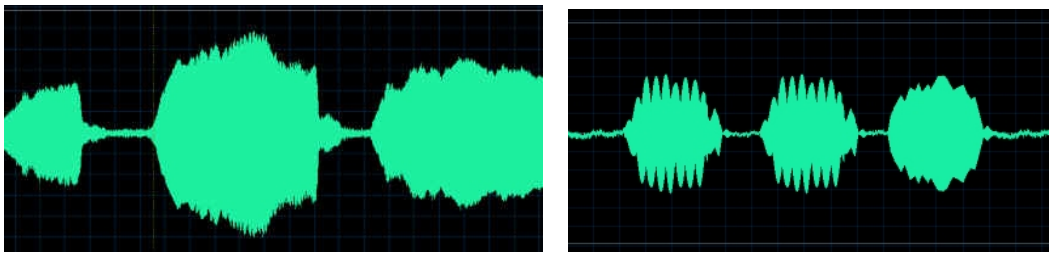


Figure 7.11: Analysis and reconstruction of a bird sound.

Definition 7.1. A *spectrogram* is a three-dimensional plot of the energy of the frequency content of a signal as it changes over time.

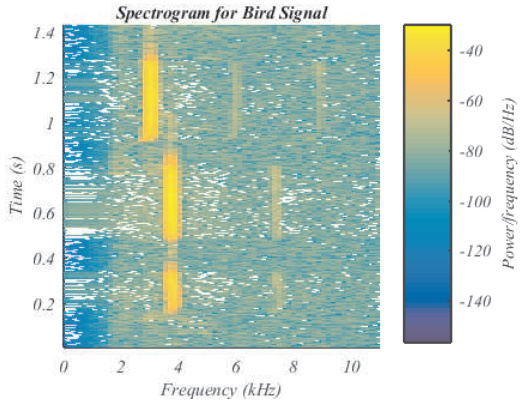


Figure 7.12: Example of spectrogram for the bird sound.

An example of a spectrogram for the bird sound in Figure 7.9 is provided in Figure 7.12. This figure was created using MATLAB's built-in function (in the Signal Processing Toolbox):

```
[y,NS,NBITS]=wavread('firstbird.wav');
spectrogram(y,128,120,128,NS);
title('Spectrogram for Bird Signal')
ylabel('Time (s)')
```

The spectrogram is created using what is called the short-time Fourier transform, or STFT. This function divides a long signal into smaller blocks, or windows, and then computes the Fourier transform on each block. This allows one to track the changes in the spectrum content over time. In Figure 7.12 one can see three different blobs in the 3kHz-4kHz range at different times, indicating how the three chirps of the bird can be picked up. This gives more information than a Fourier analysis over the entire record length.

Definition 7.2. The *power spectrum* is a plot of the portion of a signal's power (energy per unit time) falling within given frequency bins. We can either plot the Fourier coefficients, or the modulus of the Fourier transform.

Definition 7.3. Plots of $c_n = \sqrt{a_n^2 + b_n^2}$ or c_n^2 vs frequency are sometimes called *periodograms*.

An example of a periodogram for the bird sound in Figure 7.9 is provided in Figure 7.13. A periodogram can be created using MATLAB's built-in function (in the Signal Processing Toolbox):

```
[y,NS,NBITS]=wavread('firstbird.wav');
periodogram(y,[],'onesided',1024,NS)
```

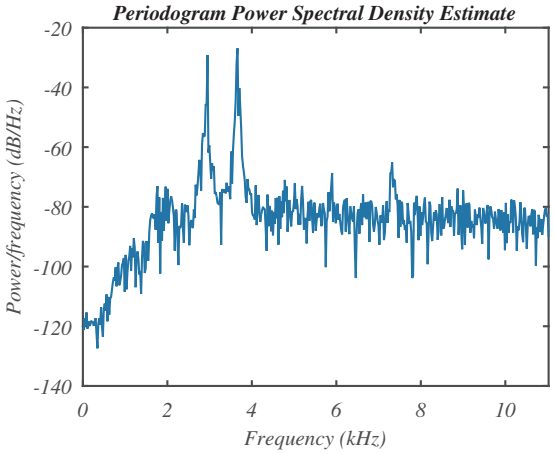


Figure 7.13: Example of periodogram for the bird sound.

There are many other types of applications. We have had students studying the oscillations of mass-spring systems and vibrating beams in differential equations. The setups are shown in Figure 7.14. On the left is a mass-spring system situated above a motion probe. The data is collected using an interface to a handheld computer. (More recently pocket PCs and other mobile devices have been used.) On the left is a similar setup for a clamped two-meter stick, which is clamped at different positions and the motion of end of the stick is monitored.

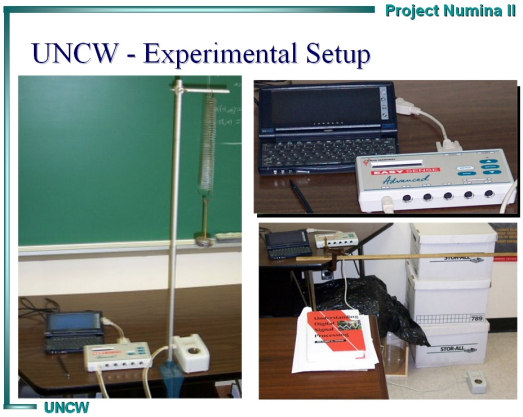


Figure 7.14: Setup for experiments for oscillations of mass-spring systems and vibrating beams. Data is recorded at a 50 Hz sampling rate using handheld devices connected to distance probes.

Of course, a simple mass on a spring exhibits the typical almost pure sinusoidal function as shown in Figure 7.15. The data is then exported to another program for analysis.

Students would learn how to fit their data to sinusoidal functions and then determine the period of oscillation as compared to the theoretical

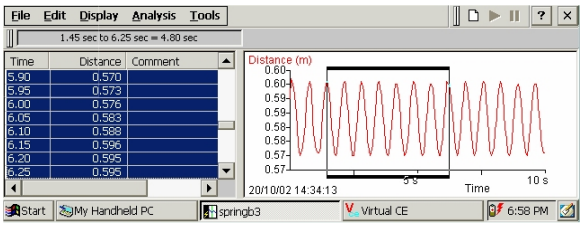


Figure 7.15: Distance vs time plot for a mass undergoing simple harmonic motion.

value. They could either do the fits in Maple (Figure 7.16) or Excel (Figure 7.17).

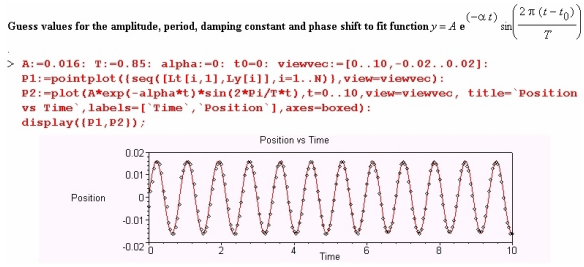


Figure 7.16: Example of fitting data in Maple.

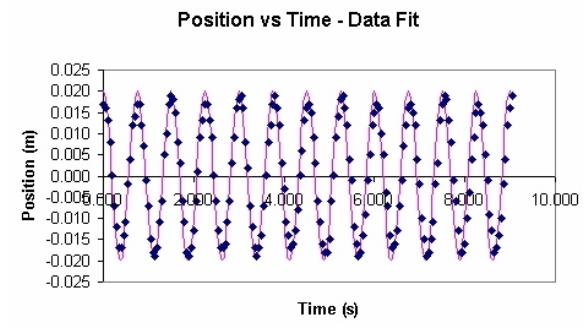


Figure 7.17: Example of fitting data in Excel.

Fitting data to damped oscillations, such as shown in Figure 7.18, is more difficult. This is the type of data one gets when measuring the vertical position of a vibrating beam using the setup shown in Figure 7.14.

Typically, one has to try to guess several parameters in order to determine the correct period, amplitude and damping. Of course, we know that it is probably better to put such a function into a program like MATLAB and then to perform a Fourier analysis on it to pick out the frequency. We did this for the signal shown in Figure 7.19. The result of the spectral analysis is shown in Figure 7.21. Do you see any predominant frequencies? Is this a better method than trying to fit the data by hand?

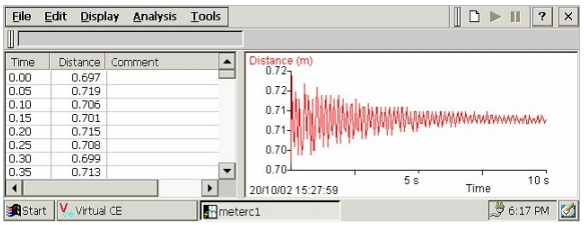
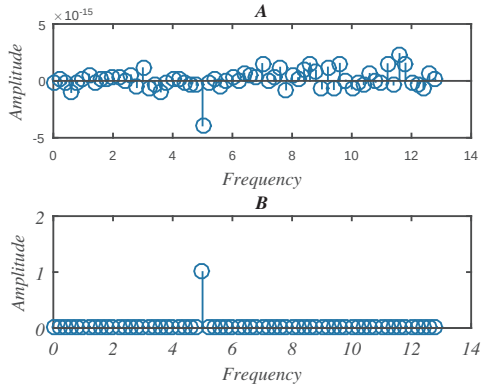


Figure 7.18: Distance vs time plot for a vibrating beam clamped at one end. The motion appears to be damped harmonic motion.

7.3 *Effects of Sampling*

We are interested in how well the discrete Fourier transform works with real signals. In the last section we saw a few examples of how signal analysis might be used. We will look into other applications later. For now, we want to examine the effects of discretization on signals so that we can make sense out of the analysis we might do on real signals. We need to begin with the simplest signals and then employ the DFT MATLAB program in Section 6.8.1 to show the results of small changes to the data.



Example 7.1. Sample the function $y(t) = \sin(10\pi t)$ for $t \in [0,5]$, and $N = 128$.

We begin by inputting the signal, $y(t) = \sin(2\pi f_0 t)$, for frequency $f_0 = 5.0$ Hz We sample this function for $N = 128$ points on the interval $[0,5]$. The Fourier Trigonometric coefficients are given in Figure 7.20. Note that the A_n's are negligible (on the order of 10^{-15}). There is a spike at 5.0 Hz. We can also plot the periodogram as shown in Figure 7.22. We again obtain the expected result of a spike at $f = 5.0$ Hz. We reconstruct the signal as shown in Figure 7.23. There appears to be agreement between the function $y(t)$, indicated by the line plot, and the reconstruction, indicated by the circles.

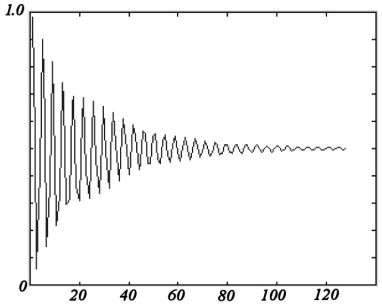


Figure 7.19: Distance vs time plot in MATLAB for a vibrating beam clamped at one end.

Figure 7.20: Fourier coefficients for the signal $y(t) = \sin(2\pi f_0 t)$ with $f_0 = 5.0$ Hz, $N = 128$, on $[0,5]$.

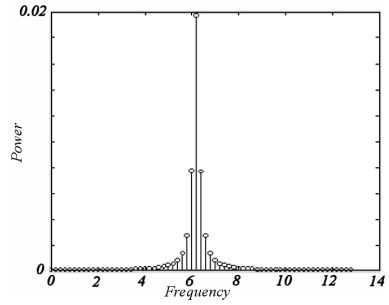


Figure 7.21: Fourier spectrum for the signal shown in Figure 7.19.

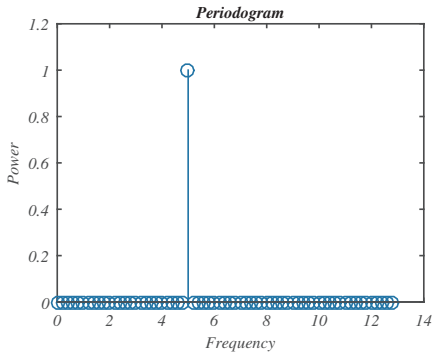


Figure 7.22: Fourier spectrum for the signal $y(t) = \sin(2\pi f_0 t)$ with $f_0 = 5.0$ Hz, $N = 128$, on $[0, 5]$.

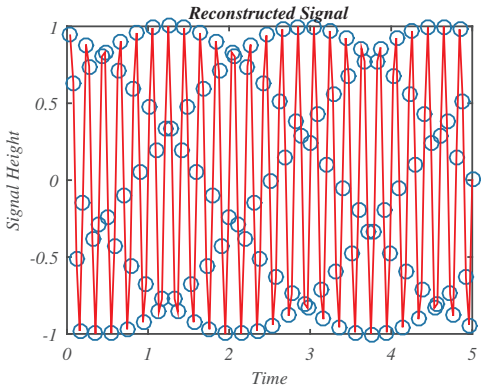


Figure 7.23: The function $y(t)$ is indicated by the line plot and the reconstruction by circles.

Example 7.2. Sample the function $y(t) = 2\sin(12\pi t) - \cos(12\pi t)$ for $t \in [0, 5]$, and $N = 128$.

In the set of Figures 7.24 and 7.25 we show the Fourier coefficients and periodogram for the function $y(t) = 2\sin(2\pi f_0 t) - \cos(2\pi f_1 t)$ for $f_0 = f_1 = 6$ Hz. We note that the heights in Figure 7.24 are the amplitudes of the sine and cosine functions. The "peaks" are located at the correct frequencies of 6 Hz. However, in the periodogram there is no information regarding the phase shift; i.e., there is no information as to whether the frequency content arises from a sine or a cosine function. We just know that all of the signal energy is concentrated at one frequency.

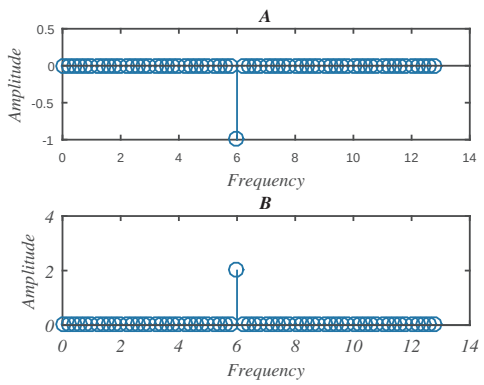


Figure 7.24: Fourier coefficients for the signal $y(t) = 2\sin(2\pi f_0 t) - \cos(2\pi f_1 t)$ with $f_0 = f_1 = 6.0$ Hz, $N = 128$, on $[0, 5]$.

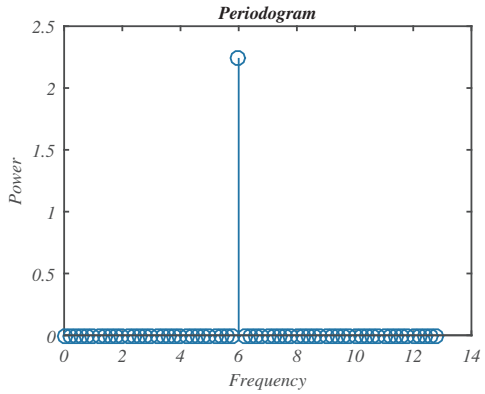


Figure 7.25: Fourier spectrum for the signal $y(t) = 2\sin(2\pi f_0 t) - \cos(2\pi f_1 t)$ with $f_0 = f_1 = 6.0$ Hz, $N = 128$, on $[0, 5]$.

Example 7.3. Sample the function $y(t) = 2\sin(12\pi t) - \cos(10\pi t)$ for $t \in [0, 5]$, and $N = 128$.

In the set of Figures 7.26 and 7.27 we show the Fourier coefficients and periodogram for the function $y(t) = 2\sin(2\pi f_0 t) - \cos(2\pi f_1 t)$ for $f_0 = 6$ Hz and $f_1 = 10$ Hz. Once again we see that the amplitudes of the Fourier coefficients are of the right height and in the right location. In the periodogram we see that the energy of the signal is distributed between two frequencies.

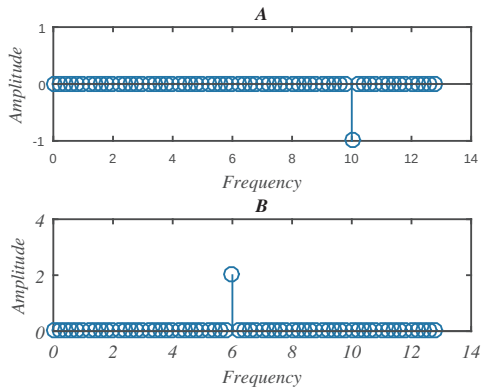


Figure 7.26: Fourier coefficients for the signal $y(t) = 2\sin(2\pi f_0 t) - \cos(2\pi f_1 t)$ with $f_0 = 6$ Hz and $f_1 = 10$ Hz, $N = 128$, on $[0, 5]$.

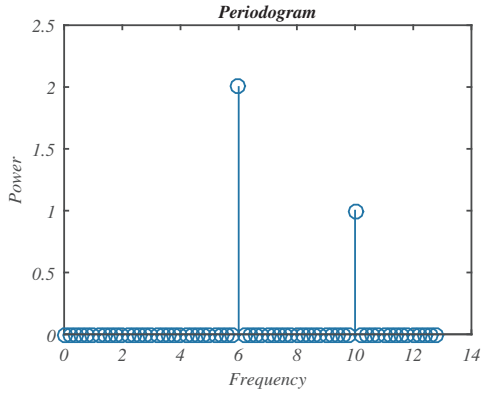


Figure 7.27: Fourier spectrum for the signal $y(t) = 2\sin(2\pi f_0 t) - \cos(2\pi f_1 t)$ with $f_0 = 6$ Hz and $f_1 = 10$ Hz. The signal was sampled with $N = 128$ points on an interval of $[0, 5]$.

Example 7.4. Sample the function $y(t) = 2\sin(12\pi t) - \cos(10\pi t)$ for $t \in [0, 5]$, and $N = 256$.

In the last several examples we have computed the spectra in using a sampled signal "recorded" over times in the interval $[0, 5]$ and sampled with $N = 128$ points. Sampling at $N = 256$ points leads to the periodogram in Figure 7.28. We note that increasing the number of points leads to a longer interval in frequency space.

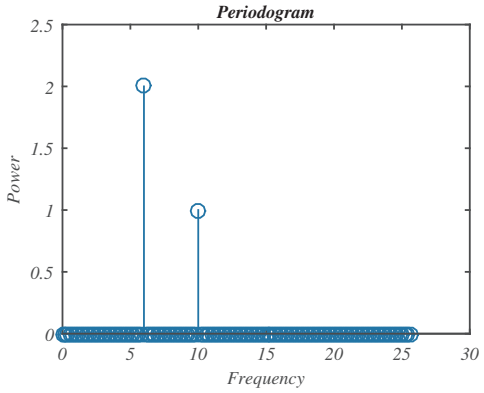


Figure 7.28: Fourier spectrum for the signal $y(t) = 2\sin(2\pi f_0 t) - \cos(2\pi f_1 t)$ with $f_0 = 6$ Hz and $f_1 = 10$ Hz. The signal was sampled with $N = 256$ points on an interval of $[0, 5]$.

Example 7.5. Sample the function $y(t) = 2\sin(12\pi t) - \cos(10\pi t)$ for $t \in [0, 10]$, and $N = 128$.

In this case we set $N = 128$ as before, but we double the record length to $T = 10$. In Figure 7.29 we see that the frequency interval has become shorter. We not only lost the 10 Hz frequency, but now we have picked up a 2.8 Hz frequency. We know that the simple signal did not have a frequency term corresponding to 2.8 Hz. So, where did this come from? Also, we note that the interval between displayed frequencies has changed.

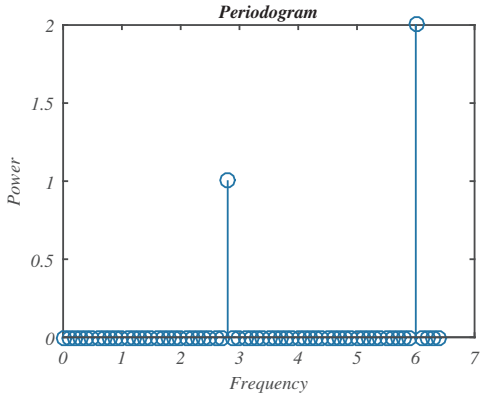


Figure 7.29: Fourier spectrum for the signal $y(t) = 2\sin(2\pi f_0 t) - \cos(2\pi f_1 t)$ with $f_0 = 6$ Hz and $f_1 = 10$ Hz. The signal was sampled with $N = 128$ points on an interval of $[0, 10]$.

For the cases where $T = 5$ the frequency spacing is 0.2 Hz as seen in Figure 7.29. However, when we increase T to 10 s, we get a frequency spacing of 0.1 Hz. Thus, it appears that $\Delta f = \frac{1}{T}$. This makes sense,

because at the beginning of the last chapter we defined

$$\omega_p = 2\pi f_p = \frac{2\pi}{T} p.$$

This gives $f_p = \frac{p}{T}$, or $f_p = 0, \frac{1}{T}, \frac{2}{T}, \frac{3}{T}, \ldots$. Thus, $\Delta f = \frac{1}{T}$. So, for $T = 5$, $\Delta f = 1/5 = 0.2$ and for $T = 10$, $\Delta f = 1/10 = 0.1$.

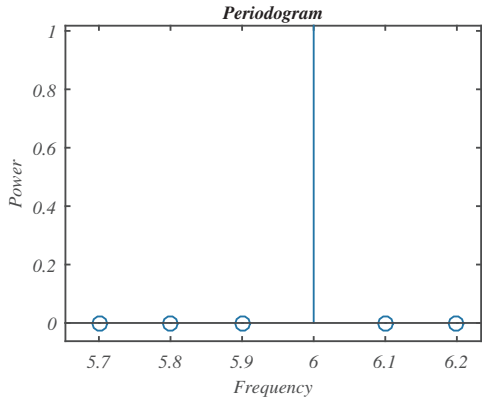


Figure 7.30: Zoomed in view of Figure 7.29.

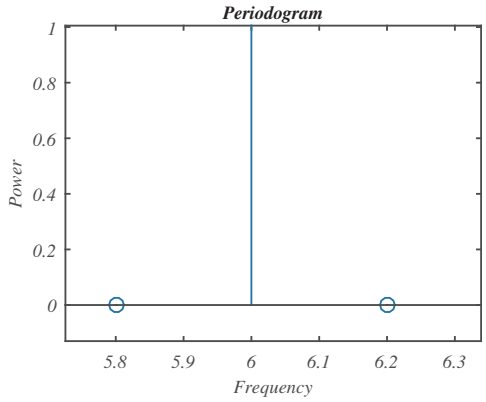


Figure 7.31: Zoomed in view of Figure 7.27. A similar view results for Figure 7.28.

So, changing the record length will change the frequency spacing. But why does changing T introduce frequencies that are not there? What if we reduced N? We saw that increasing N leads to longer frequency intervals. Will reducing it lead to a problem similar to increasing T? In Figure 7.32 we see the result of using only 64 points. Yes, again we see the occurrence of a 2.8 Hz spike. Also, the range of displayed frequencies is shorter. So, the range of displayed frequencies depends upon both the numbers of points at which the signal is sampled and the record length.

We will explain this masquerading of frequencies in terms of something called aliasing. However, that is not the whole story. Notice that the frequencies represented in the periodograms is discrete. Even in the case that $T = 5$ and $N = 128$, we only displayed frequencies at intervals of $\frac{1}{T} = 0.2$. What would happen if the signal had a frequency in between these values?

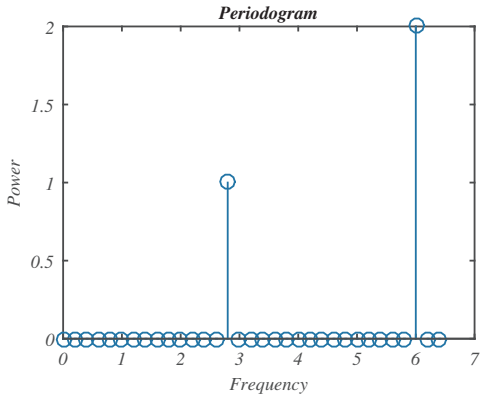


Figure 7.32: Fourier spectrum for the signal $y(t) = 2\sin(2\pi f_0 t) - \cos(2\pi f_1 t)$ with $f_0 = 6$ Hz and $f_1 = 10$ Hz. The signal was sampled with $N = 64$ points on an interval of $[0, 5]$.

Example 7.6. Sample the function $y(t) = \sin(12.2\pi t)$ for $t \in [0, 5]$, and $N = 256$.

In this example we consider changing the the 6 Hz frequency to 6.1 Hz. We see the result in Figure 7.33. Since we could not pinpoint the signal's frequencies at one of the allowed discrete frequencies, the periodogram displays a spread in frequencies. This phenomenon is called *ringing* or *spectral leakage*.

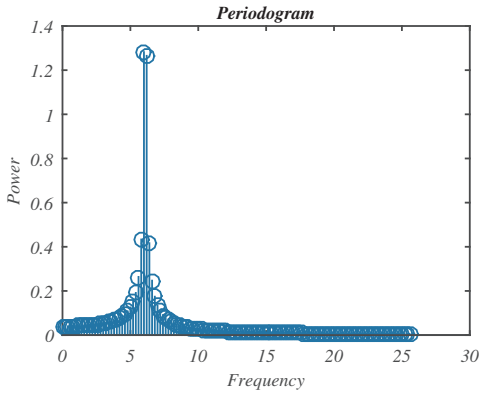


Figure 7.33: Fourier spectrum for the signal $y(t) = 2\sin(2\pi f_0 t)$ with $f_0 = 6.1$ Hz. The signal was sampled with $N = 256$ points on an interval of $[0, 5]$.

It is also interesting to see the effects on the individual Fourier coefficients. This is shown in Figure 7.34. While there is some apparent distribution of energy amongst the A_n's, it is still essentially zero. Most of the effects indicate that the energy is distributed amongst sine contributions.

What we have learned from these examples is that we need to be careful in picking the record length and number of samples used in analyzing analog signals. Sometimes we have control over these parameters, but other times we are stuck with them depending upon the limitations of the recording devices. Next we will investigate how the effects of ringing and aliasing occur.

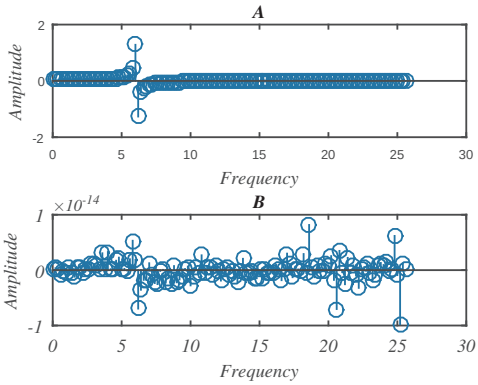


Figure 7.34: Fourier spectrum for the signal $y(t) = 2\sin(2\pi f_0 t)$ with $f_0 = 6.1$ Hz. The signal was sampled with $N = 256$ points on an interval of $[0, 5]$.

7.4 *Effect of Finite Record Length*

In the previous section we saw examples of the effects of finite record length on the spectrum of sampled data. In order to understand these effects for general signals, we will focus on a signal containing only one frequency, such as $y(t) = \sin(2\pi f_0 t)$. We will record this signal over a finite time interval, $t \in [0, T]$. This leads us to studying a finite wave train. (Recall that we has seen examples of finite wave trains earlier in the chapter on Fourier Transforms and in the last chapter. However, in these cases we used a cosine function. Also, in one of these cases we integrated over a symmetric interval.)

We will consider sampling the finite sine wave train given by

$$y(t) = \begin{cases} \sin 2\pi f_0 t, & 0 \le t \le T \\ 0, & \text{otherwise} \end{cases} . \tag{7.1}$$

In order to understand the spectrum of this signal, we will first compute the Fourier transform of this function. Afterwards, we will show how sampling this finite wave train affects the Fourier transform.

Example 7.7. Compute the Fourier transform of the finite sine wave train.

We begin by computing the Fourier transform of the finite wave train and write the transform in terms of its real and imaginary parts. The computation is straightforward and we obtain

$$\begin{aligned} \hat{y}(f) &= \int_{-\infty}^{\infty} y(t) e^{2\pi i f t}\, dt \\ &= \int_0^T \sin(2\pi f_0 t)\cos(2\pi f t)\, dt + i\int_0^T \sin(2\pi f_0 t)\sin(2\pi f t)\, dt \\ &= \frac{1}{2}\int_0^T \left[\sin(2\pi(f+f_0)t) - \sin(2\pi(f-f_0)t)\right] dt \\ &\quad + \frac{i}{2}\int_0^T \left[\cos(2\pi(f-f_0)t) - \cos(2\pi(f+f_0)t)\right] dt \\ &= \frac{1}{4\pi}\left[\frac{1}{f+f_0} - \frac{1}{f-f_0}\right] \end{aligned}$$

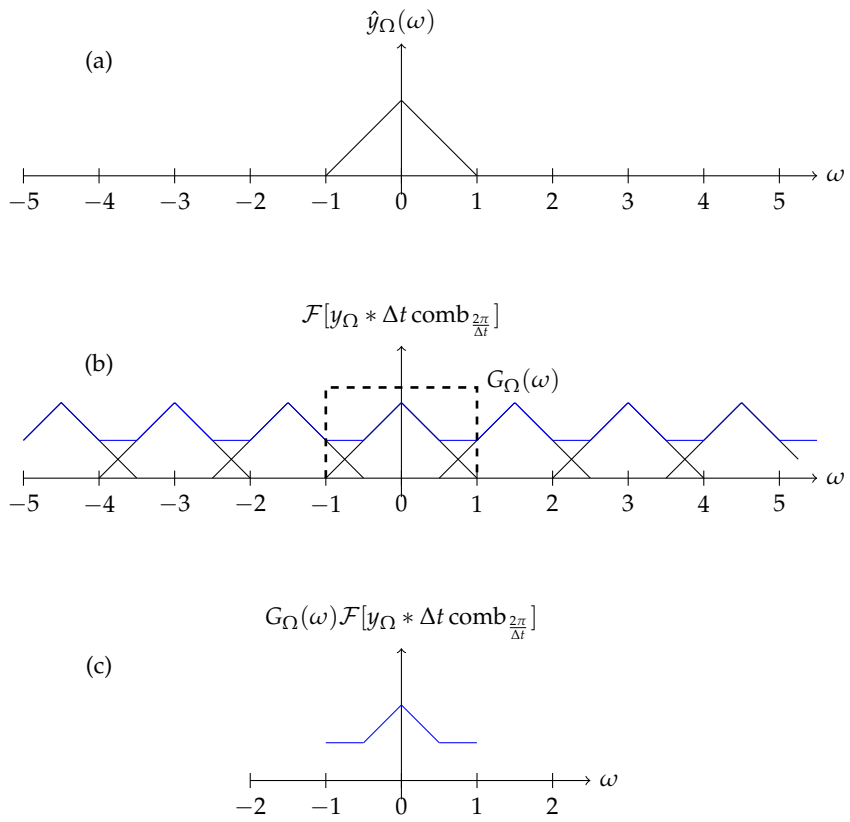


Figure 7.55: Sampling a bandlimited function, $y_\Omega(t)$, where $\Omega = 1$ and $\Delta t = \frac{4\pi}{3}$. (a) $\hat{y}_\Omega(\omega)$ is a triangular function with bandwidth $2\Omega = 2$. (b) A picture of $\mathcal{F}[y_\Omega * \Delta t \operatorname{comb}_{\frac{2\pi}{\Delta t}}]$ with the gate function. (c) The spectrum of the sampled signal, $G_\Omega(\omega)\mathcal{F}[y_\Omega * \Delta t \operatorname{comb}_{\frac{2\pi}{\Delta t}}]$.

Sketch the function $G_\Omega(\omega)\mathcal{F}[y_\Omega * \Delta t \operatorname{comb}_{\frac{2\pi}{\Delta t}}]$ for sampling time $\Delta t = \frac{4\pi}{3}$.

Correct sampling would be done with $\Delta t = \pi$. So, we expect overlapping copies in the Fourier transform of the sampled signal. In Figure 7.55 we show the process.

In Figure 7.55(a) we show the sketch of the triangular function, $\hat{y}_\Omega(\omega)$. The bandwith is $2\Omega = 2$.

Next, in Figure 7.55(b) we draw the translations of $\hat{y}_\Omega(\omega)$ in multiples of $2\pi/\Delta t = 1.5$. Superimposed on the translations are the sum, represented by $\mathcal{F}[y_\Omega * \Delta t \operatorname{comb}_{\frac{2\pi}{\Delta t}}]$.

The Fourier transform of the sampled signal is then obtained by multiplying the sum of the translations in Figure 7.55(b) by $G_\Omega(\omega)$ to obtain the final result, $G_\Omega(\omega)\mathcal{F}[y_\Omega * \Delta t \operatorname{comb}_{\frac{2\pi}{\Delta t}}]$. This is shown in Figure 7.55(c).

7.7 *Nonstationary Signals*

7.7.1 *Simple examples*

A MAJOR ASSUMPTION MADE IN USING Fourier transforms is that the frequency content of a signal does not change in time. Such signals are called

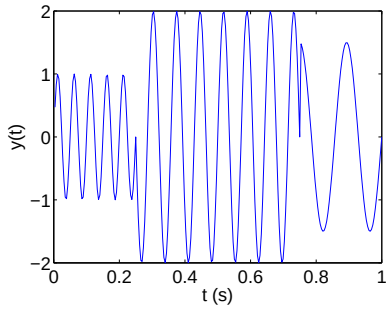


Figure 7.56: A plot of the function $f(t)$ vs t.

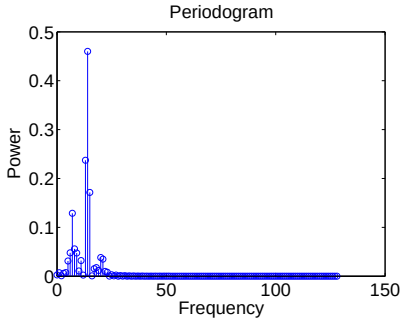


Figure 7.57: The application of the DFT algorithm to $f(t)$.

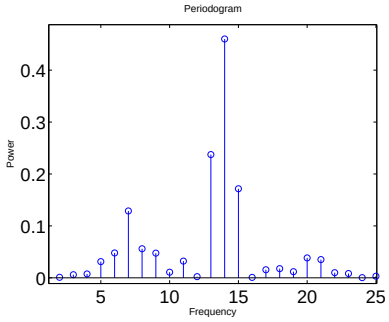


Figure 7.58: A magnified view of the DFT of $f(t)$.

stationary. Consider the following example.

Example 7.14. Let

$$f(t) = \begin{cases} 1\sin(2\pi f_0 t), & 0 \le t < 0.25, < \\ 2\sin(2\pi f_1 t), & 0.25 < t < 0.75, \\ 1.5\sin(2\pi f_2 t), & 0.75 < t \le 1, \end{cases}$$

where $f_0 = 20$ Hz, $f_1 = 14$ Hz and $f_3 = 7$ Hz.

A plot of $f(t)$ is shown in Figure 7.56. There are three frequencies present, but occur at during different time intervals. The spectrum of this signal using the discrete Fourier transform over the entire time interval using $N = 256$ gives the plot in Figure 7.56. It indicates many more frequencies are present than just the three we know about. In Figure 7.58 we show a blow-up of the region containing the largest values. While it looked like there might have been ringing, there are peaks at the main frequencies, but somehow we could not capture the fact that the signal is nonstationary.

We can capture the time dependence of the frequency content by splitting the time series into four blocks of width 0.25. This is shown in Figure 7.59. Now we apply the DFT to each block as shown in Figure 7.60. Zooming in further in Figure 7.61, we see that each periodogram displays different frequency content. However, since each block of $f(t)$ is not a perfect sine function, there still is a little inaccuracy in picking out the exact frequency in each block.

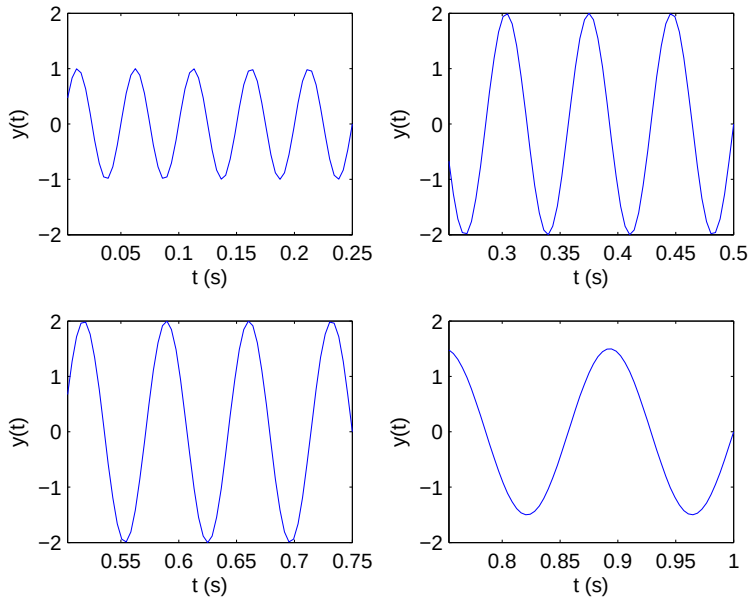


Figure 7.59: A plot of the function $f(t)$ vs t split into four windows.

Example 7.15. Consider changing the time interval to $[0, 4\pi]$ in the

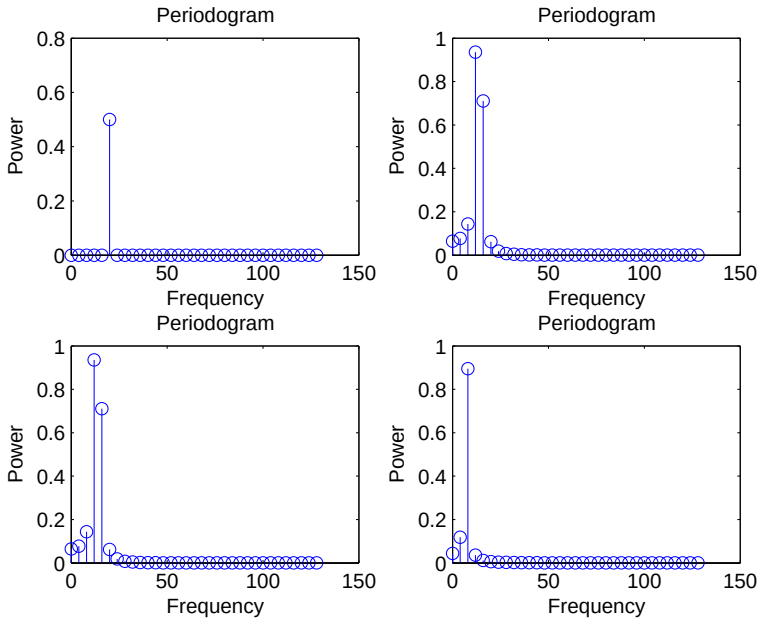


Figure 7.60: The application of the DFT algorithm to each of the four blocks of $f(t)$.

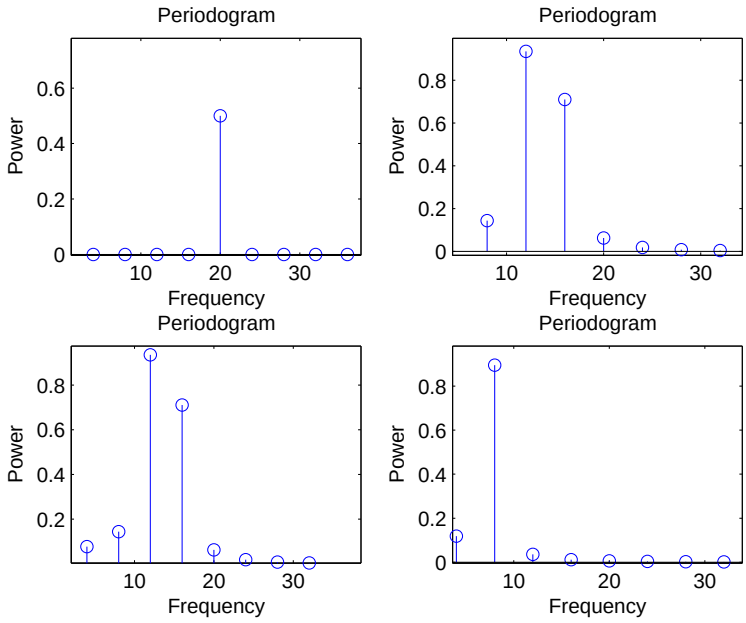


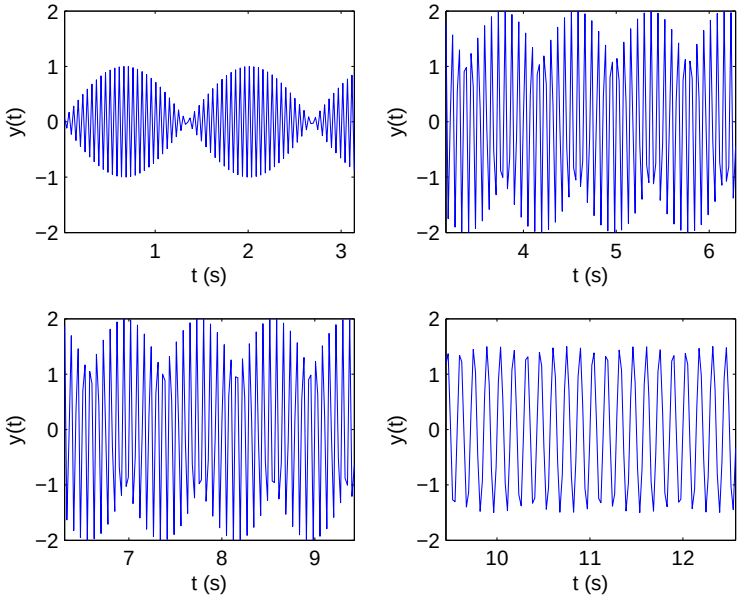
Figure 7.61: A magnified view of the DFT of the four blocks of $f(t)$.

previous example. Let

$$g(t) = \begin{cases} \sin(2\pi f_0 t), & 0 \le t < \pi, \\ 2\sin(2\pi f_1 t), & \pi \le t < 3\pi, \\ 1.5\sin(2\pi f_2 t), & 3\pi \le t \le 4\pi, \end{cases}$$

where $f_0 = 20$ Hz, $f_1 = 14$ Hz and $f_3 = 7$ Hz. The four blocks are shown in Figure 7.62. The DFT for these blocks with $N = 512$ is shown in Figure 7.63. We see that the frequencies are more defined and correct for the most part.

Figure 7.62: The four blocks of $g(t)$.



Another example is the "chirp" function. A chirp is a sinusoidal function with a time varying frequency. When turned into a sound of the right length and frequency range, a chirp sounds like the chirp of a bird. a linear chirp is one in which the frequency changes linearly. The next example gives an example of a chirp.

Example 7.16. Consider the linear chirp signal

$$y(t) = \sin(2\pi \left(f_0 + (f_1 - f_0)\frac{t}{T})t \right), \quad t \in [0, 1],$$

for $f_0 = 1.0$ Hz and $f_1 = 10.0$ Hz.

The function takes the form $y(t) = \sin 2\pi f t$, where the frequency is time-dependent, $f(t) = f_0 + (f_1 - f_0)\frac{t}{T}$. In Figure 7.64 we show this linear chirp. The frequency varies from $f_0 = 1.0$ Hz to $f_1 = 10.0$ Hz. When one computes the DTF of this signal, the periodogram in Figure 7.65 results. As one can see, a variety of frequencies appear and there is no indication that the frequency is time varying.

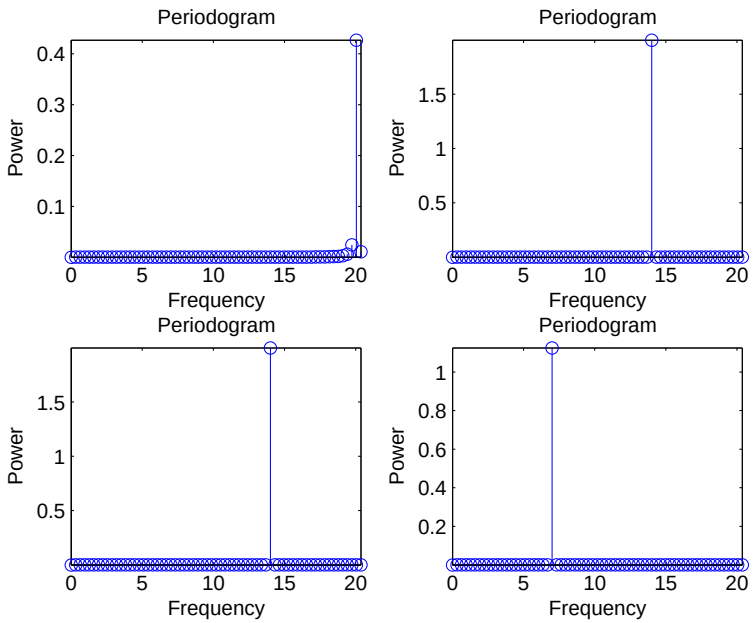


Figure 7.63: A magnified view of the DFT of the four blocks of $g(t)$.

7.7.2 *The Spectrogram*

The examples in the last section point to the need for a modification of the Fourier transform for analog signals and the DFT for discrete signals. In Example 7.15, we saw that computing the DFT over subintervals of the signal, we can attempt to find the time dependence of the frequency spectrum. This idea can be generalized for both continuous and discrete Fourier transforms. In MATLAB the function **spectrogram** produces a plot of the time-dependent frequency of a signal by using similar blocks, but by sliding blocks of a given width in time across the signal and doing a Fourier analysis for each block. The output is a spectrogram.

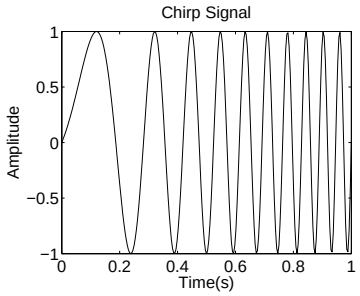


Figure 7.64: A chirp signal.

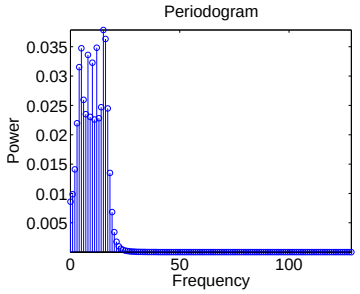


Figure 7.65: DFT of the chirp signal.

Example 7.17. Let

$$g(t) = \begin{cases} \sin(2\pi f_0 t), & 0 \le t < \pi, \\ 2\sin(2\pi f_1 t), & \pi \le t < 3\pi, \\ 1.5\sin(2\pi f_2 t), & 3\pi \le t \le 4\pi, \end{cases}$$

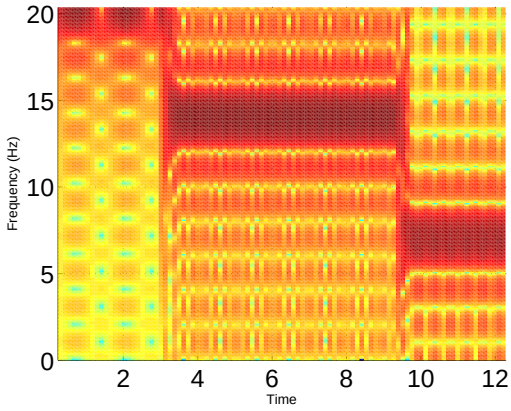
where $f_0 = 20$ Hz, $f_1 = 14$ Hz and $f_3 = 7$ Hz.

We sample this signal at 512 points and using the MATLAB command `spectrogram`, in the form

```
spectrogram(g,rectwin(20),15,n,1/dt,'yaxis')
```

to generate the spectrogram in Figure 7.66, where **y** is the signal sampled at **n** $= 512$ points and **dt** $= 4\pi/n$. The blocks are 20 pts wide with an overlap of 15 points. Note that the dominant three frequencies appear roughly at the correct locations.

Figure 7.66: The spectrogram plot of $g(t)$.



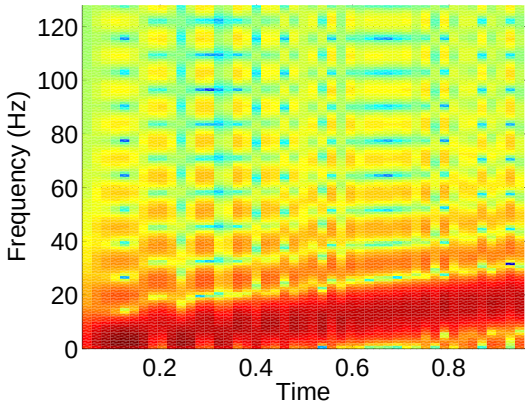
Example 7.18. Let $y(t) = \sin(2\pi\left(f_0 + (f_1 - f_0)\frac{t}{T}\right)t)$, $t \in [0,1]$, for $f_0 = 1.0$ Hz and $f_1 = 10.0$ Hz.

The the spectrogram in Figure 7.67 was produced by sampling this signal at 512 points and using the MATLAB command

```
spectrogram(y,rectwin(20),15,n,1/dt,'yaxis')
```

In the lower part of the figure we see a fuzzy linear region indicating the linear dependence of the frequency on time roughly going from $f = 1$ Hz to $f = 10.$ Hz. The resolution of this frequency content depends partly on the width of the rectangular block used and the overlap of the blocks. In this case the command takes the form **spectrogram(y,rectwin(w),o)**, where **w** is the width and **o** is the size of the overlap.

Figure 7.67: The spectrogram plot of the chirp $y(t) = \sin(2\pi\left(f_0 + (f_1 - f_0)\frac{t}{T}\right)t)$, $t \in [0,1]$, for $f_0 = 1.0$ Hz and $f_1 = 10.0$ Hz.



Let's see how to formalize the process described in the last examples. We begin with a sampled signal, y_n, $n = 0, \ldots, N-1$, and a rectangular window (or, block), $w_n = w(t_n)$ of width $M < N$. Then, we compute the transform of the product,

$$Y_{k,\ell} = DFT\left\{[y_\ell w_0, \ldots, y_{\ell+n} w_n, \ldots, y_{\ell+N-1} w_{N-1}]\right\}$$

This gives the spectrum which we can associate with a time associated with a time over which the block is nonzero. Next, the window is translated by a time t_ℓ. The shifted window is given by $w_{n-\ell} = w(t_n - t_\ell)$.

Example 7.19. A simple example of using overlapping blocks.

We consider the signal

$$y(t) = \begin{cases} 2\sin 2\pi t, & 0 \le t \le 5, \\ 1.5\sin 3\pi t, & 5 \le t \le 10. \end{cases}$$

This is shown in Figure 7.68(a).

The signal is then sampled with $\Delta t = 0.05$ as shown in Figure 7.68(b).

Figure 7.68(c) shows the blocks that can be used. Each block is of width 1.0 and translated y 0.75, leaving an overlap of 0.25 between consecutive blocks.

We can change the values of the width of the rectangular block used and the overlap of the blocks to see the effects on the output. Examples are provided in Figures 7.69-7.70 for Examples 7.15 and 7.16. Each row is a spectrogram of fixed with with a 20%, 40% 60% or 80% blockwidth overlap as the blocks are translated across the signal. The block widths down the figure are 10 pts, 20 pts, 30 pts, and 40 pts, respectively.

In Figures 7.69-7.70 we see that there is better frequency resolution for wider blocks, but the time resolution is blurrier. However, increasing the overlap with aid in resolving the time as well. The better frequency resolution is due to using more points in the DFT for that block.

7.7.3 Short-Time Fourier Transform

THE KEY TO STUDYING NONSTATIONARY SIGNALS is the Short-Time Fourier Transform (STFT). The continuous version of the discrete Short-Time Fourier Transform is obtained by multiplying the signal by a sliding window function, $w(t)$, which is translated translated along the time axis, and taking the Fourier transform. The idea of Short-Time Fourier Transform is often credited to Dennis Gabor's work in 1946. He used a Gaussian window function.

Formally, we define the window function, $w(t)$, and multiply a shifted window, $w(t-\tau)$, by the signal, $y(t)$, and compute

$$STFT[y](\tau,\omega) \equiv= Y(\tau,\omega) = \int_{-\infty}^{\infty} y(t)w(t-\tau)e^{-i\omega t}\,dt. \tag{7.17}$$

Note that the sign in the exponential is negative, which differs from our earlier convention.

For the purpose of computation we can discretize the time and frequency variables giving the discrete-time Short-Time Fourier Transform and the discrete Short-Time Fourier Transform. The discrete-time Short-Time Fourier Transform is given by

$$Y(n,\omega) = \sum_{m=-\infty}^{\infty} x[m]w[m-n]e^{-i\omega n}. \tag{7.18}$$

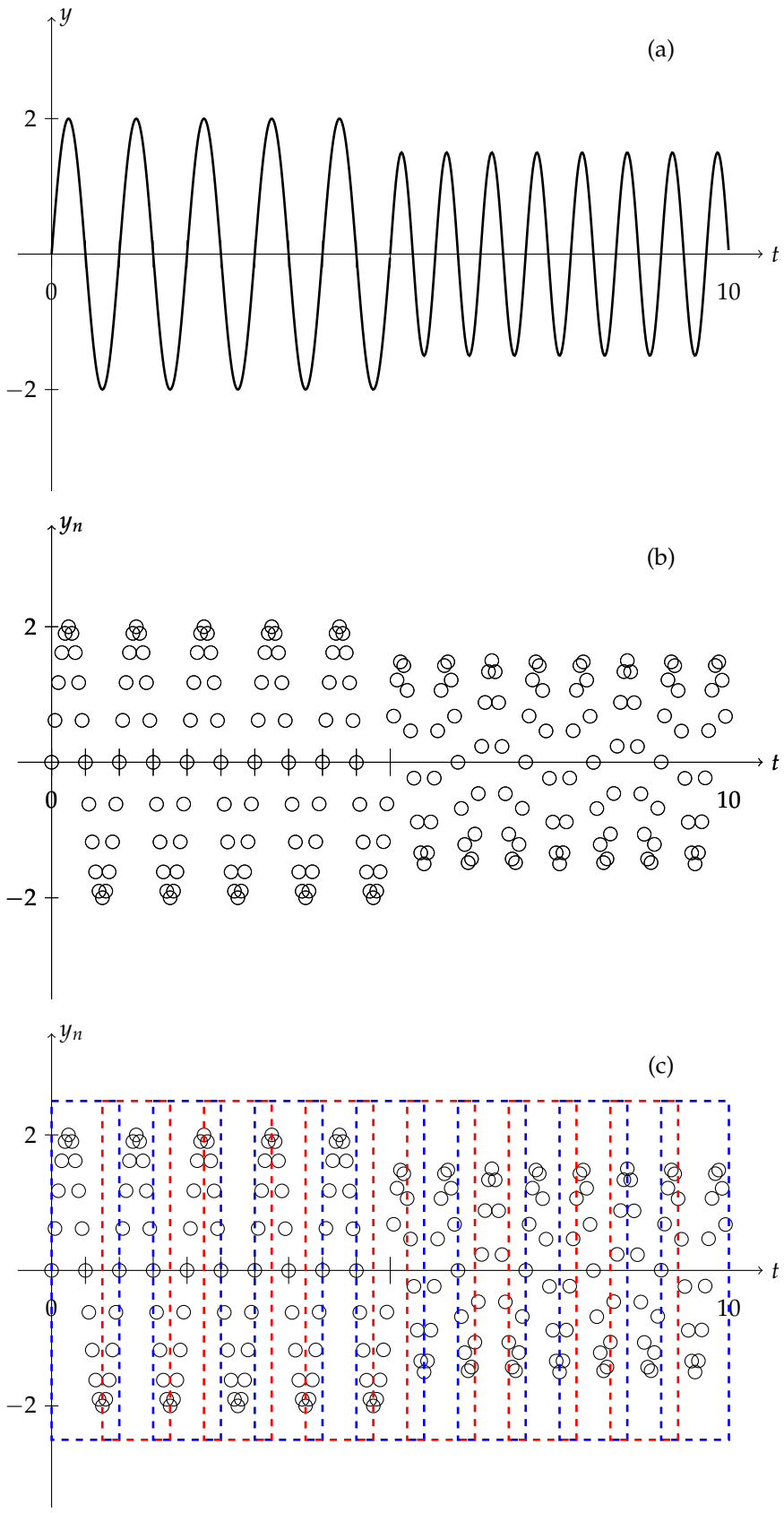


Figure 7.68: (a) Plot of $y(t)$. (b) Sampled signal y_n. (c) Translated windows showing overlapping.

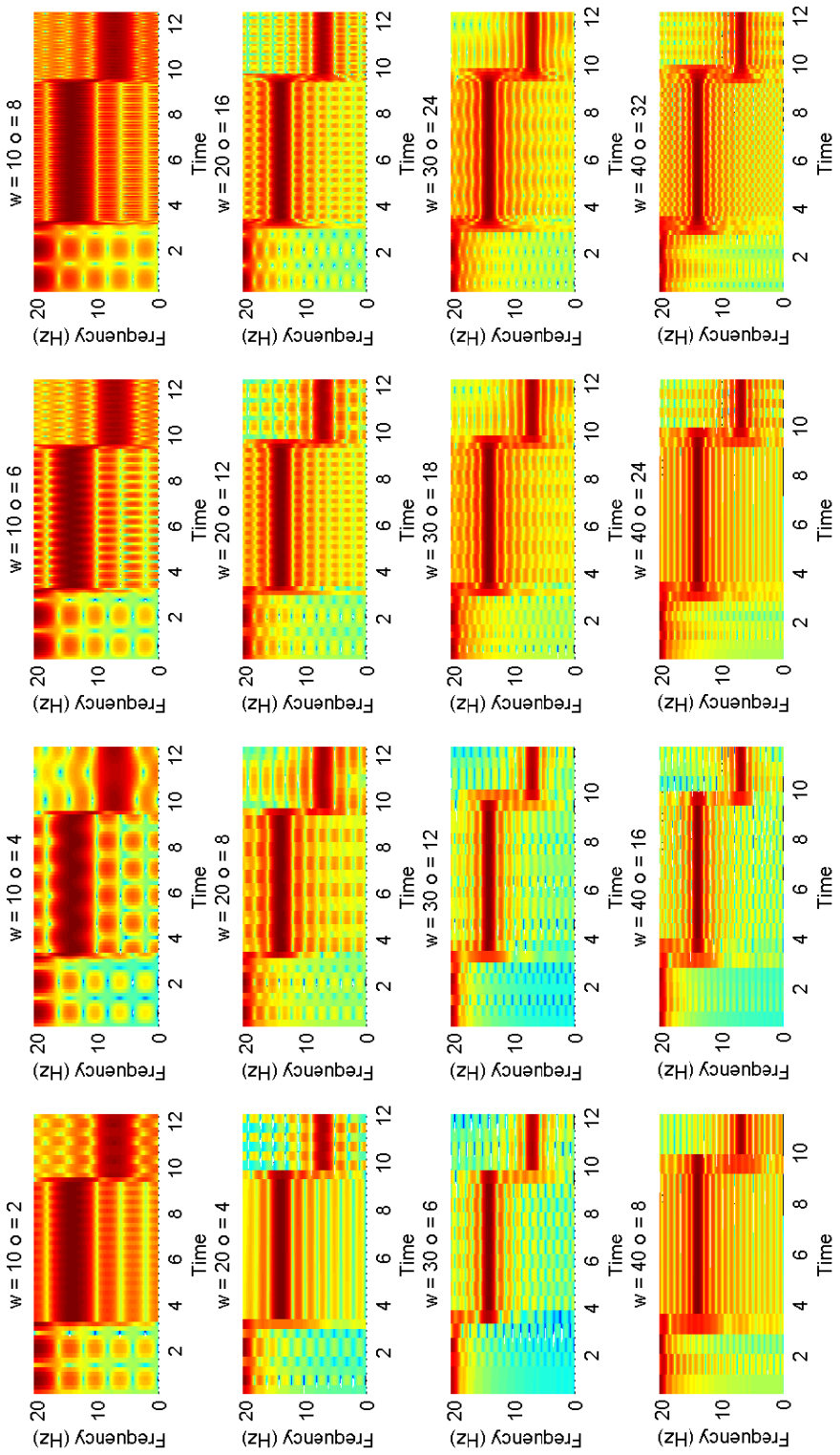


Figure 7.69: The spectrogram plot of $g(t)$ in Example 7.15.

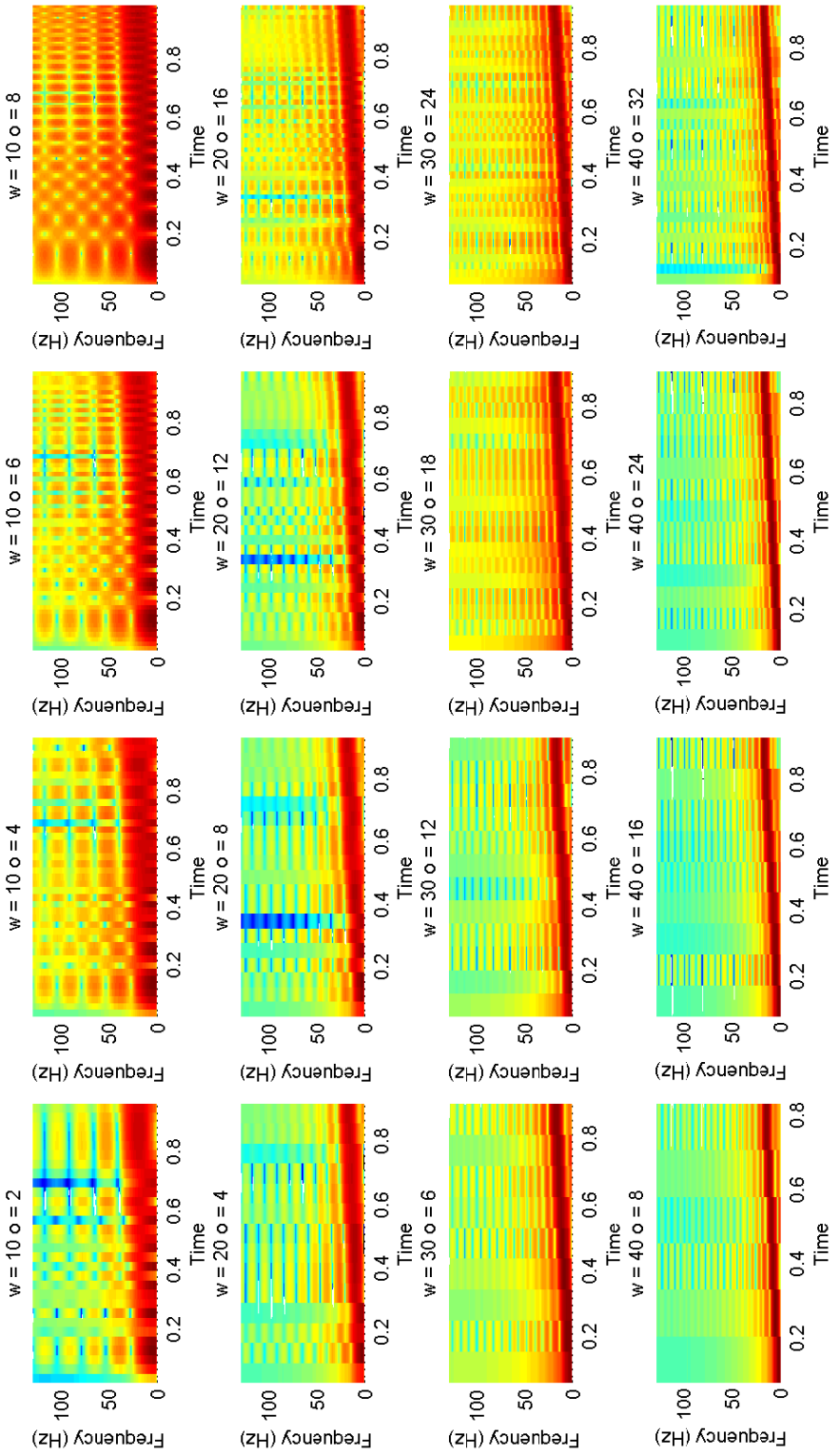


Figure 7.70: The spectrogram plot of the chirp signal, $f(t)$, in Example 7.16.

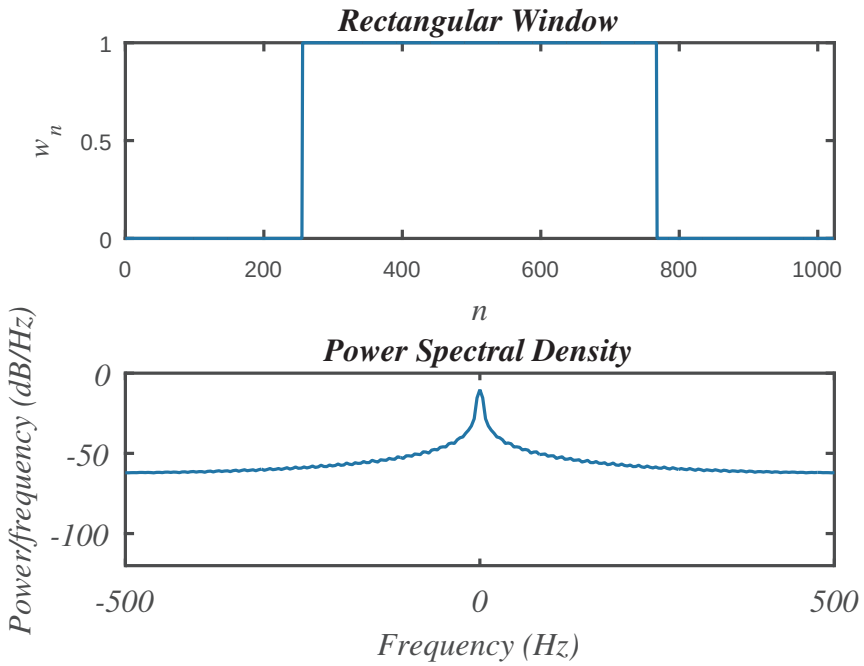


Figure 7.72: A plot of the rectangular window with $a = 0.25$ and its Power Spectral Density in dB/Hz.

This window, named after Maurice S. Bartlett (1910-2002) is not used in practice, but its Fourier transform is easily computed. The resulting plot of the triangular window and its Power Spectral Density in dB/Hz is shown in Figure 7.73. Here we plot

```
n=0:N-1;
w=1-abs(n-(N-1)/2)/((N-1)/2);
```

We see that is has a behavior similar to that of the rectangular window. However, the spectral leakage is less. A more general form of the Bartlett window is given by

$$w(t) = \begin{cases} 1 - \frac{2|t|}{T}, & |t| < \frac{T}{2}, \\ 0, & |t| \geq \frac{T}{2}. \end{cases}$$

A family of bell-shaped windows can be obtained in the form

$$w(t) = a + b\cos(2\pi f t).$$

This is known as a generalized Hamming window, named after Richard Wesley Hamming (1915-1998). Two members of this family are the Hamming window,

$$w(t) = \begin{cases} 0.538 + 0.461\cos\frac{2\pi t}{T}, & |t| < \frac{T}{2}, \\ 0, & |t| \geq \frac{T}{2}, \end{cases}$$

and the Hann Window (named after Julius Ferdinand von Hann (1839-1921)),

$$w(t) = \begin{cases} 0.5 + 0.5\cos\frac{2\pi t}{T}, & |t| < \frac{T}{2}, \\ 0, & |t| \geq \frac{T}{2}. \end{cases}$$

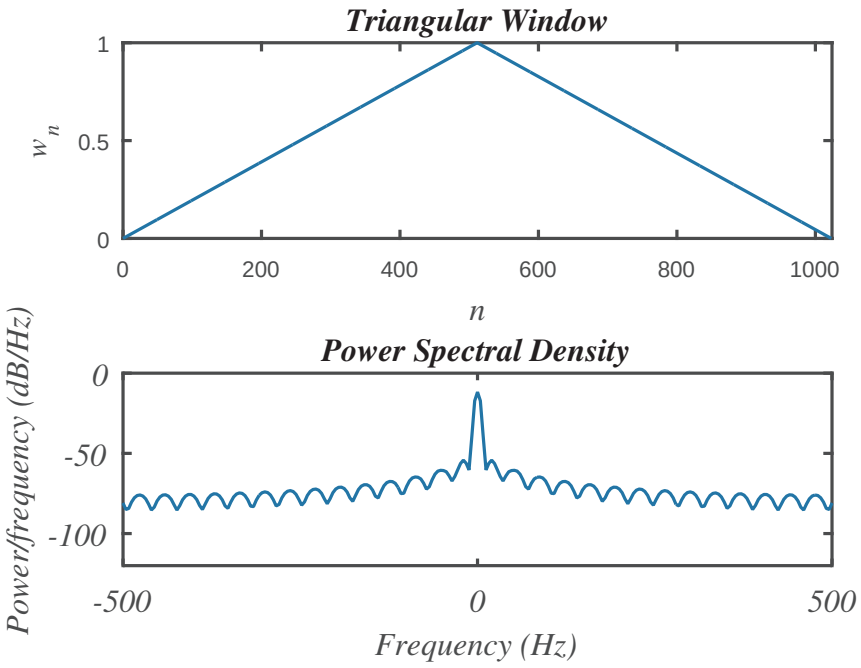


Figure 7.73: A plot of the triangular window and its Power Spectral Density in dB/Hz.

Another common window is the Blackman Window, named after Ralph Beebe Blackman (1904-1990) .

$$w(t) = \begin{cases} 0.427 + 0.497\cos\frac{2\pi t}{T} + 0.0768\cos\frac{4\pi t}{T}, & |t| < \frac{T}{2}, \\ 0, & |t| \geq \frac{T}{2}. \end{cases}$$

In practice, one defines the windows at a set of discrete times. The Hamming and Hann window can be encoded as

$$w[n] = 0.5\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right), \quad n = 0, 1, \ldots, N-1$$

and the Hann window,

$$w[n] = a - b\cos\left(\frac{2\pi n}{N-1}\right), \quad n = 0, 1, \ldots, N-1,$$

where

$$a = \frac{25}{46} \approx 0.53836, \quad b = 1 - a \approx 0.46164.$$

The plots of the Hamming and Hann windows and their Spectral Density plots are shown in Figures 7.74-7.75, repsectively.

In Figure 7.76 we show a comparison of the Power Spectral Density plots for the rectangular window as compared to a typical plot for the other windows. We see that there are small differences in the latter three, but all provide less leakage effects than the rectangular window. In particular, the width of the central peaks is smaller and the heights of the sidelobes are smaller.

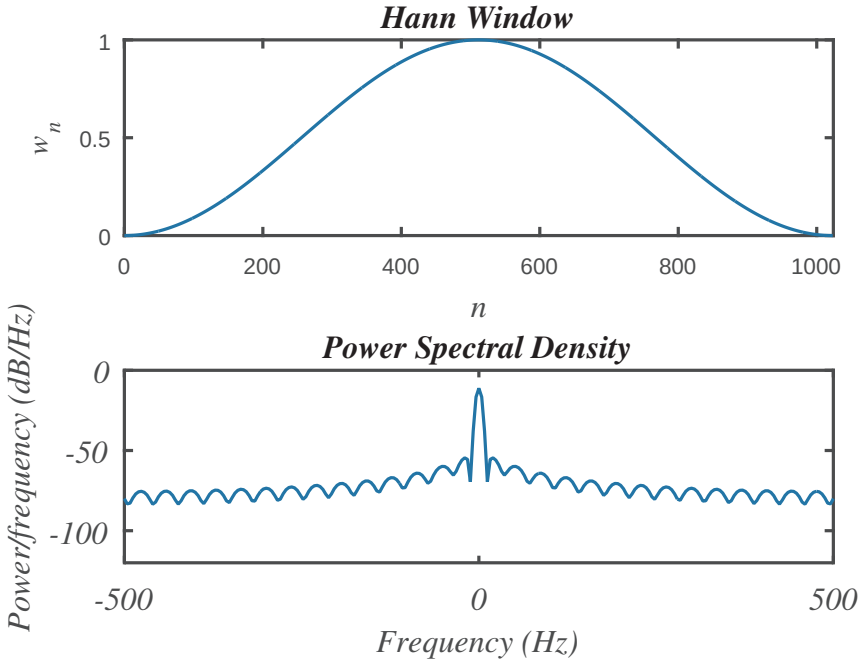


Figure 7.74: A plot of the Hann window and its Power Spectral Density in dB/Hz.

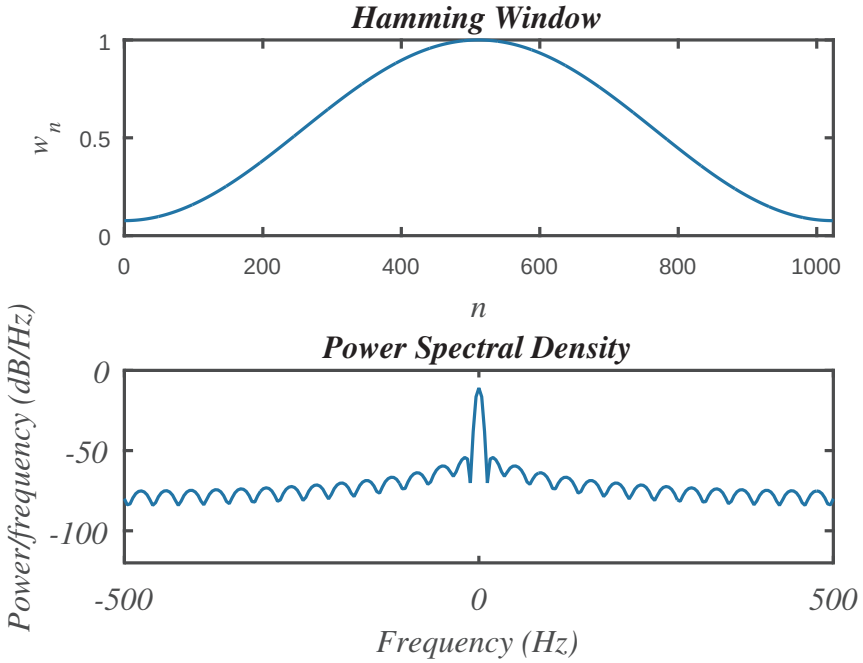


Figure 7.75: A plot of the Hamming window and its Power Spectral Density in dB/Hz.

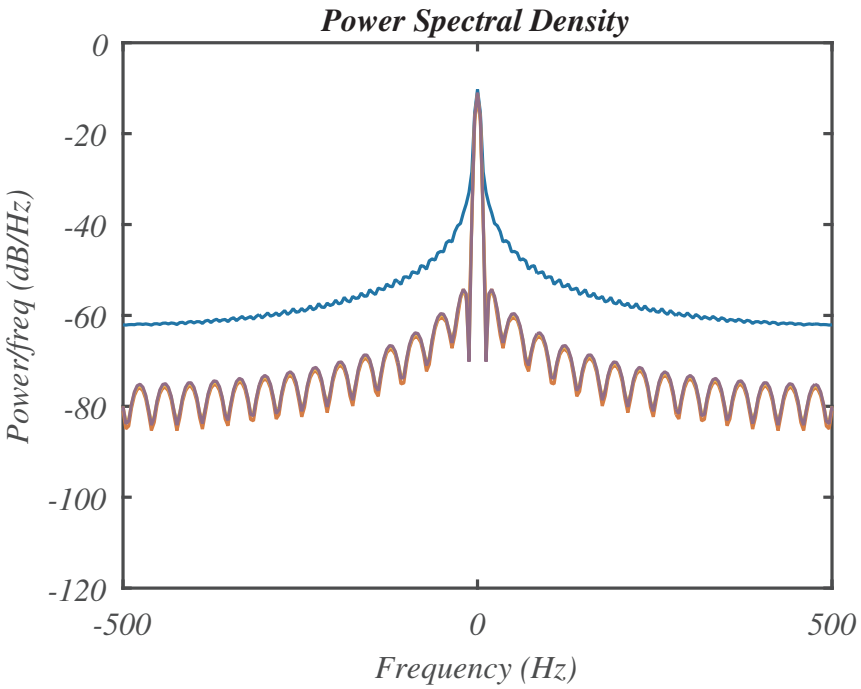


Figure 7.76: A comparison of the Power Spectral Density plots for the rectangular window as compared to a typical plot for the other windows discussed in the text. Note how the rectangular window leakage is larger than the others.

These figures were created using the MATLAB code below.

```
clear
close all
Fs=1024;
T=1;
N=T*Fs;
n=0:N-1;
fg=0;
for irn=1:4
    fg=fg+1;
    switch irn
        case 1
            figure(fg)
            nz=.25*N;
            w1=[zeros(1,nz),ones(1,N-2*nz),zeros(1,nz)];
            subplot(211)
            plot(n,w1)
            title('Rectangular Window')
            xlabel('n')
            ylabel('w_n')
            axis([0 N 0 1])
            subplot(212)
            pwelch(w1,[],[],[],Fs,'centered');
            axis([-500 500 -120 0])
        case 2
            figure(fg)
```

```
            w2=1-abs(n-(N-1)/2)/((N-1)/2);
            subplot(211)
            plot(n,w2)
           title('Triangular Window')
            xlabel('n')
            ylabel('w_n')
            axis([0 N 0 1])
            subplot(212)
            pwelch(w2,[],[],[],Fs,'centered');
            axis([-500 500 -120 0])
        case 3
            figure(fg)
            w3=(1-cos(2*pi*n/(N-1)))/2;
            subplot(211)
            plot(n,w3)
            title('Hann Window')
            xlabel('n')
            ylabel('w_n')
            axis([0 N 0 1])
            subplot(212)
            pwelch(w3,[],[],[],Fs,'centered');
            axis([-500 500 -120 0])
        case 4
            figure(fg)
            alpha=0.53836;
            beta=1-alpha;
            w4=alpha-beta*cos(2*pi*n/(N-1));
            subplot(211)
            plot(n,w4)
            title('Hamming Window')
            xlabel('n')
            ylabel('w_n')
            axis([0 N 0 1])
            subplot(212)
            pwelch(w4,[],[],[],Fs,'centered');
            axis([-500 500 -120 0])
    end
end

[pxx1,f1]=pwelch(w1,[],[],[],Fs,'centered');
[pxx2,f2]=pwelch(w2,[],[],[],Fs,'centered');
[pxx3,f3]=pwelch(w3,[],[],[],Fs,'centered');
[pxx4,f4]=pwelch(w4,[],[],[],Fs,'centered');

figure(5)
plot(f1,10*log10(pxx1),f2,10*log10(pxx2), ...
```

```
     f3,10*log10(pxx3),f4,10*log10(pxx4))
axis([-500 500 -120 0])
xlabel('Frequency (Hz)')
ylabel('Power/freq (dB/Hz)')
title('Power Spectral Density')
legend('Rectangular','Triangular','Hann','Hamming')

figure(6)
plot(f1,10*log10(pxx1),f2,10*log10(pxx2), ...
     f3,10*log10(pxx3),f4,10*log10(pxx4))
axis([-100 100 -120 0])
xlabel('Frequency (Hz)')
ylabel('Power/freq (dB/Hz)')
title('Power Spectral Density')
legend('Rectangular','Triangular','Hann','Hamming')
```

In MATLAB there are many built-in functions useful for studying the spectral behavior of window functions. For example, **wvtool** produces the Figures 7.77-7.79 which show comparisons with a rectangular window. In Figure 7.77 we show the Power Spectral Density for a rectangular window and a Hann window with $N = 128$. Lowering N to a value of 64, we can see the effect of N on the spectral behavior of the windows in Figure 7.78. In Figure Figure 7.79 we show the Power Spectral Density for a rectangular window and a Hamming window with $N = 64$. It show similar characteristics to that of the Hann window.

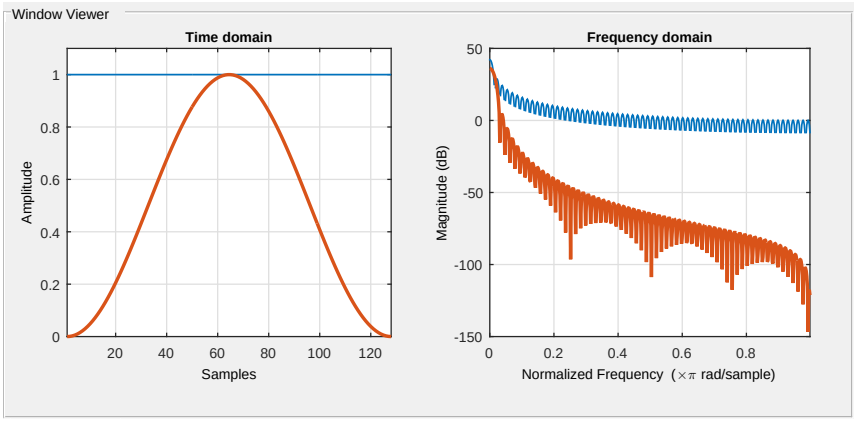


Figure 7.77: A comparison of the Power Spectral Density for a rectangular window and a Hann window with $N = 128$.

Window functions are useful for improving the spectral analysis of real signals. Using window functions in the spectral domain, we create spectral filters. These are useful in controlling the frequency content in the signal and these filters are an important topic in signal processing. We leave any detailed discussion of filtering and applications of windows to real signals for texts devoted specifically to signal processing. In this book we mainly have introduced the topic as an important application of Fourier analysis. We will conclude with one more application, that of harmonic analysis when

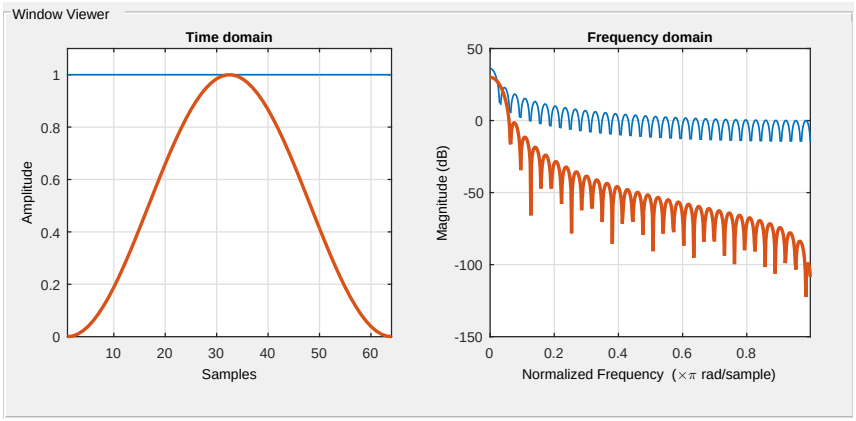


Figure 7.78: A comparison of the Power Spectral Density for a rectangular window and a Hann window with $N = 64$.

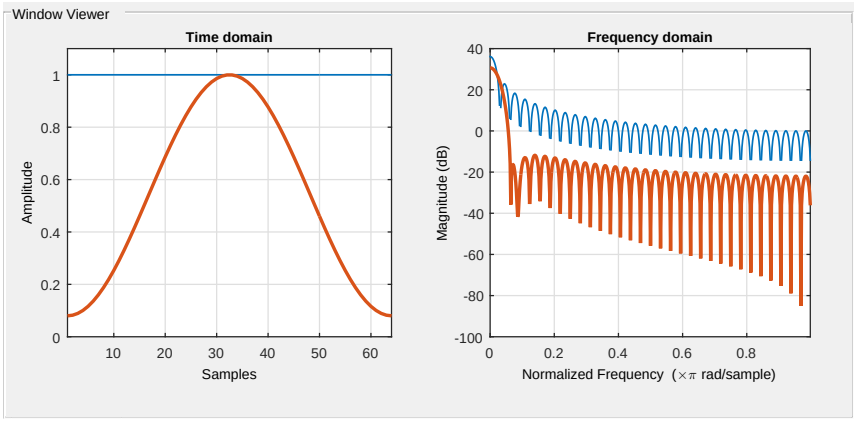


Figure 7.79: A comparison of the Power Spectral Density for a rectangular window and a Hamming window with $N = 64$.